IBM Spectrum Scale
5.1.1

Command and Programming Reference

GC28-3197-02

Note

Before using this information and the product it supports, read the information in “Notices” on page 1693.

This edition applies to Version 5 release 1 modification 1 of the following products, and to all subsequent releases and modifications until otherwise indicated in new editions:

- IBM Spectrum Scale Data Management Edition ordered through Passport Advantage® (product number 5737-F34)
- IBM Spectrum Scale Data Access Edition ordered through Passport Advantage (product number 5737-I39)
- IBM Spectrum Scale Erasure Code Edition ordered through Passport Advantage (product number 5737-J34)
- IBM Spectrum Scale Data Management Edition ordered through AAS (product numbers 5641-DM1, DM3, DM5)
- IBM Spectrum Scale Data Access Edition ordered through AAS (product numbers 5641-DA1, DA3, DA5)
- IBM Spectrum Scale Data Management Edition for IBM® ESS (product number 5765-DME)
- IBM Spectrum Scale Data Access Edition for IBM ESS (product number 5765-DAE)

Significant changes or additions to the text and illustrations are indicated by a vertical line (|) to the left of the change.

IBM welcomes your comments; see the topic “How to send your comments” on page xl. When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Tables

About this information

This edition applies to IBM Spectrum Scale version 5.1.1 for AIX®, Linux®, and Windows.

IBM Spectrum Scale is a file management infrastructure, based on IBM General Parallel File System (GPFS) technology, which provides unmatched performance and reliability with scalable access to critical file data.

To find out which version of IBM Spectrum Scale is running on a particular AIX node, enter:

```
lslpp -l gpfs\*
```

To find out which version of IBM Spectrum Scale is running on a particular Linux node, enter:

```
rpm -qa | grep gpfs     (for SLES and Red Hat Enterprise Linux)
```

```
dpkg -l | grep gpfs     (for Ubuntu Linux)
```

To find out which version of IBM Spectrum Scale is running on a particular Windows node, open **Programs and Features** in the control panel. The IBM Spectrum Scale installed program name includes the version number.

Which IBM Spectrum Scale information unit provides the information you need?

The IBM Spectrum Scale library consists of the information units listed in Table 1 on page xxii.

To use these information units effectively, you must be familiar with IBM Spectrum Scale and the AIX, Linux, or Windows operating system, or all of them, depending on which operating systems are in use at your installation. Where necessary, these information units provide some background information relating to AIX, Linux, or Windows. However, more commonly they refer to the appropriate operating system documentation.

Note: Throughout this documentation, the term "Linux" refers to all supported distributions of Linux, unless otherwise specified.

Table 1. IBM Spectrum Scale library information units		
Information unit	**Type of information**	**Intended users**
IBM Spectrum Scale: Concepts, Planning, and Installation Guide	This guide provides the following information: **Product overview** • Overview of IBM Spectrum Scale • GPFS architecture • Protocols support overview: Integration of protocol access methods with GPFS • Active File Management • AFM-based Asynchronous Disaster Recovery (AFM DR) • Introduction to AFM to cloud object storage • Data protection and disaster recovery in IBM Spectrum Scale • Introduction to IBM Spectrum Scale GUI • IBM Spectrum Scale management API • Introduction to Cloud services • Introduction to file audit logging • Introduction to clustered watch folder • Understanding call home • IBM Spectrum Scale in an OpenStack cloud deployment • IBM Spectrum Scale product editions • IBM Spectrum Scale license designation • Capacity based licensing **Planning** • Planning for GPFS • Planning for protocols • Planning for Cloud services • Planning for AFM • Planning for AFM DR • Planning for AFM to cloud object storage • Firewall recommendations • Considerations for GPFS applications • Security-Enhanced Linux support • Space requirements for call home data upload	System administrators, analysts, installers, planners, and programmers of IBM Spectrum Scale clusters who are very experienced with the operating systems on which each IBM Spectrum Scale cluster is based

Table 1. IBM Spectrum Scale library information units (continued)		
Information unit	**Type of information**	**Intended users**
IBM Spectrum Scale: Concepts, Planning, and Installation Guide	**Installing** • Steps for establishing and starting your IBM Spectrum Scale cluster • Installing IBM Spectrum Scale on Linux nodes and deploying protocols • Installing IBM Spectrum Scale on AIX nodes • Installing IBM Spectrum Scale on Windows nodes • Installing Cloud services on IBM Spectrum Scale nodes • Installing and configuring IBM Spectrum Scale management API • Installation of Active File Management (AFM) • Installing and upgrading AFM-based Disaster Recovery • Installing call home • Installing file audit logging • Installing clustered watch folder • Steps to permanently uninstall GPFS **Upgrading** • IBM Spectrum Scale supported upgrade paths • Online upgrade support for protocols and performance monitoring • Upgrading IBM Spectrum Scale nodes	System administrators, analysts, installers, planners, and programmers of IBM Spectrum Scale clusters who are very experienced with the operating systems on which each IBM Spectrum Scale cluster is based

Table 1. IBM Spectrum Scale library information units (continued)		
Information unit	**Type of information**	**Intended users**
IBM Spectrum Scale: Concepts, Planning, and Installation Guide	• Upgrading IBM Spectrum® Scale non-protocol Linux nodes • Upgrading IBM Spectrum Scale protocol nodes • Upgrading AFM and AFM DR • Upgrading object packages • Upgrading SMB packages • Upgrading NFS packages • Upgrading call home • Manually upgrading the performance monitoring tool • Manually upgrading pmswift • Manually upgrading the IBM Spectrum Scale management GUI • Upgrading Cloud services • Upgrading to IBM Cloud Object Storage software level 3.7.2 and above • Upgrade paths and commands for file audit logging and clustered watch folder • Upgrading with clustered watch folder enabled • Upgrading IBM Spectrum Scale components with the installation toolkit • Changing IBM Spectrum Scale product edition • Completing the upgrade to a new level of IBM Spectrum Scale • Reverting to the previous level of IBM Spectrum Scale	System administrators, analysts, installers, planners, and programmers of IBM Spectrum Scale clusters who are very experienced with the operating systems on which each IBM Spectrum Scale cluster is based

Table 1. IBM Spectrum Scale library information units (continued)		
Information unit	**Type of information**	**Intended users**
IBM Spectrum Scale: Concepts, Planning, and Installation Guide	• Coexistence considerations • Compatibility considerations • Considerations for IBM Spectrum Protect for Space Management • Applying maintenance to your GPFS system • Guidance for upgrading the operating system on IBM Spectrum Scale nodes • Considerations for upgrading from an operating system not supported in IBM Spectrum Scale 5.1.0.x • Servicing IBM Spectrum Scale protocol nodes • Offline upgrade with complete cluster shutdown	

Table 1. IBM Spectrum Scale library information units (continued)		
Information unit	**Type of information**	**Intended users**
IBM Spectrum Scale: Administration Guide	This guide provides the following information: **Configuring** • Configuring the GPFS cluster • Configuring the CES and protocol configuration • Configuring and tuning your system for GPFS • Parameters for performance tuning and optimization • Ensuring high availability of the GUI service • Configuring and tuning your system for Cloud services • Configuring IBM Power Systems for IBM Spectrum Scale • Configuring file audit logging • Configuring clustered watch folder • Configuring Active File Management • Configuring AFM-based DR • Configuring AFM to cloud object storage • Tuning for Kernel NFS backend on AFM and AFM DR • Configuring call home • Integrating IBM Spectrum Scale Cinder driver with Red Hat OpenStack Platform 16.1 **Administering** • Performing GPFS administration tasks • Verifying network operation with the mmnetverify command • Managing file systems • File system format changes between versions of IBM Spectrum Scale • Managing disks • Managing protocol services • Managing protocol user authentication	System administrators or programmers of IBM Spectrum Scale systems

Table 1. IBM Spectrum Scale library information units (continued)		
Information unit	**Type of information**	**Intended users**
IBM Spectrum Scale: Administration Guide	• Managing protocol data exports • Managing object storage • Managing GPFS quotas • Managing GUI users • Managing GPFS access control lists • Native NFS and GPFS • Accessing a remote GPFS file system • Information lifecycle management for IBM Spectrum Scale • Creating and maintaining snapshots of file systems • Creating and managing file clones • Scale Out Backup and Restore (SOBAR) • Data Mirroring and Replication • Implementing a clustered NFS environment on Linux • Implementing Cluster Export Services • Identity management on Windows / RFC 2307 Attributes • Protocols cluster disaster recovery • File Placement Optimizer • Encryption • Managing certificates to secure communications between GUI web server and web browsers • Securing protocol data • Cloud services: Transparent cloud tiering and Cloud data sharing • Managing file audit logging • RDMA tuning • Configuring Mellanox Memory Translation Table (MTT) for GPFS RDMA VERBS Operation • Administering AFM • Administering AFM DR	System administrators or programmers of IBM Spectrum Scale systems

Table 1. IBM Spectrum Scale library information units (continued)		
Information unit	**Type of information**	**Intended users**
IBM Spectrum Scale: Administration Guide	• Administering AFM to cloud object storage • Highly-available write cache (HAWC) • Local read-only cache • Miscellaneous advanced administration • GUI limitations	System administrators or programmers of IBM Spectrum Scale systems

Table 1. IBM Spectrum Scale library information units (continued)		
Information unit	**Type of information**	**Intended users**
IBM Spectrum Scale: Problem Determination Guide	This guide provides the following information: **Monitoring** • Performance monitoring • Monitoring system health through the IBM Spectrum Scale GUI • Monitoring system health by using the mmhealth command • Monitoring events through callbacks • Monitoring capacity through GUI • Monitoring AFM and AFM DR • Monitoring AFM to cloud object storage • GPFS SNMP support • Monitoring the IBM Spectrum Scale system by using call home • Monitoring remote cluster through GUI • Monitoring file audit logging • Monitoring clustered watch • Monitoring local read-only cache **Troubleshooting** • Best practices for troubleshooting • Understanding the system limitations • Collecting details of the issues • Managing deadlocks • Installation and configuration issues • Upgrade issues • Network issues • File system issues • Disk issues • Security issues • Protocol issues • Disaster recovery issues • Performance issues	System administrators of GPFS systems who are experienced with the subsystems used to manage disks and who are familiar with the concepts presented in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*

Table 1. IBM Spectrum Scale library information units (continued)		
Information unit	**Type of information**	**Intended users**
IBM Spectrum Scale: Problem Determination Guide	• GUI and monitoring issues • AFM issues • AFM DR issues • AFM to cloud object storage issues • Transparent cloud tiering issues • File audit logging issues • Troubleshooting mmwatch • Maintenance procedures • Recovery procedures • Support for troubleshooting • References	

Table 1. IBM Spectrum Scale library information units (continued)		
Information unit	**Type of information**	**Intended users**
IBM Spectrum Scale: Command and Programming Reference	This guide provides the following information: **Command reference** • gpfs.snap command • mmaddcallback command • mmadddisk command • mmaddnode command • mmadquery command • mmafmconfig command • mmafmcosaccess command • mmafmcosconfig command • mmafmcosctl command • mmafmcoskeys command • mmafmctl command • mmafmlocal command • mmapplypolicy command • mmaudit command • mmauth command • mmbackup command • mmbackupconfig command • mmbuildgpl command • mmcachectl command • mmcallhome command • mmces command • mmcesdr command • mmchattr command • mmchcluster command • mmchconfig command • mmchdisk command • mmcheckquota command • mmchfileset command • mmchfs command • mmchlicense command • mmchmgr command • mmchnode command • mmchnodeclass command • mmchnsd command • mmchpolicy command • mmchpool command • mmchqos command • mmclidecode command	• System administrators of IBM Spectrum Scale systems • Application programmers who are experienced with IBM Spectrum Scale systems and familiar with the terminology and concepts in the XDSM standard

Table 1. IBM Spectrum Scale library information units (continued)		
Information unit	**Type of information**	**Intended users**
IBM Spectrum Scale: Command and Programming Reference	• mmclone command • mmcloudgateway command • mmcrcluster command • mmcrfileset command • mmcrfs command • mmcrnodeclass command • mmcrnsd command • mmcrsnapshot command • mmdefedquota command • mmdefquotaoff command • mmdefquotaon command • mmdefragfs command • mmdelacl command • mmdelcallback command • mmdeldisk command • mmdelfileset command • mmdelfs command • mmdelnode command • mmdelnodeclass command • mmdelnsd command • mmdelsnapshot command • mmdf command • mmdiag command • mmdsh command • mmeditacl command • mmedquota command • mmexportfs command • mmfsck command • mmfsctl command • mmgetacl command • mmgetstate command • mmhadoopctl command • mmhdfs command • mmhealth command • mmimgbackup command • mmimgrestore command • mmimportfs command • mmkeyserv command	• System administrators of IBM Spectrum Scale systems • Application programmers who are experienced with IBM Spectrum Scale systems and familiar with the terminology and concepts in the XDSM standard

Table 1. IBM Spectrum Scale library information units (continued)		
Information unit	**Type of information**	**Intended users**
IBM Spectrum Scale: Command and Programming Reference	• mmlinkfileset command • mmlsattr command • mmlscallback command • mmlscluster command • mmlsconfig command • mmlsdisk command • mmlsfileset command • mmlsfs command • mmlslicense command • mmlsmgr command • mmlsmount command • mmlsnodeclass command • mmlsnsd command • mmlspolicy command • mmlspool command • mmlsqos command • mmlsquota command • mmlssnapshot command • mmmigratefs command • mmmount command • mmnetverify command • mmnfs command • mmnsddiscover command • mmobj command • mmperfmon command • mmpmon command • mmprotocoltrace command • mmpsnap command • mmputacl command • mmqos command • mmquotaoff command • mmquotaon command • mmreclaimspace command • mmremotecluster command • mmremotefs command • mmrepquota command • mmrestoreconfig command • mmrestorefs command • mmrestripefile command	• System administrators of IBM Spectrum Scale systems • Application programmers who are experienced with IBM Spectrum Scale systems and familiar with the terminology and concepts in the XDSM standard

Table 1. IBM Spectrum Scale library information units (continued)		
Information unit	**Type of information**	**Intended users**
IBM Spectrum Scale: Command and Programming Reference	• mmrestripefs command • mmrpldisk command • mmsdrrestore command • mmsetquota command • mmshutdown command • mmsmb command • mmsnapdir command • mmstartup command • mmtracectl command • mmumount command • mmunlinkfileset command • mmuserauth command • mmwatch command • mmwinservctl command • spectrumscale command **Programming reference** • IBM Spectrum Scale Data Management API for GPFS information • GPFS programming interfaces • GPFS user exits • IBM Spectrum Scale management API endpoints • Considerations for GPFS applications	• System administrators of IBM Spectrum Scale systems • Application programmers who are experienced with IBM Spectrum Scale systems and familiar with the terminology and concepts in the XDSM standard

Table 1. IBM Spectrum Scale library information units (continued)		
Information unit	**Type of information**	**Intended users**
IBM Spectrum Scale: Big Data and Analytics Guide	This guide provides the following information: Summary of changes Hadoop Scale Storage Architecture • Elastic Storage Server (ESS) • Erasure Code Edition • Share Storage (SAN-based storage) • File Placement Optimizer (FPO) • Deployment model • Additional supported storage features IBM Spectrum Scale support for Hadoop • HDFS transparency overview • Supported IBM Spectrum Scale storage modes • Hadoop cluster planning • CES HDFS • Non-CES HDFS • Security • Advanced features • Hadoop distribution support • Limitations and differences from native HDFS • Problem determination IBM Spectrum Scale Hadoop performance tuning guide • Overview • Performance overview • Hadoop Performance Planning over IBM Spectrum Scale • Performance guide	• System administrators of IBM Spectrum Scale systems • Application programmers who are experienced with IBM Spectrum Scale systems and familiar with the terminology and concepts in the XDSM standard

Table 1. IBM Spectrum Scale library information units (continued)		
Information unit	**Type of information**	**Intended users**
IBM Spectrum Scale: Big Data and Analytics Guide	Cloudera Data Platform (CDP) Private Cloud Base • Overview • Planning • Installation • Configuration • Administration • Limitations • Problem determination	• System administrators of IBM Spectrum Scale systems • Application programmers who are experienced with IBM Spectrum Scale systems and familiar with the terminology and concepts in the XDSM standard
IBM Spectrum Scale: Big Data and Analytics Guide	Cloudera HDP 3.X • Planning • Installation • Upgrading and uninstallation • Configuration • Administration • Limitations • Problem determination Open Source Apache Hadoop • Open Source Apache Hadoop without CES HDFS • Open Source Apache Hadoop with CES HDFS Cloudera HDP 2.6 • Planning • Installation • Upgrading software stack • Configuration • Administration • Troubleshooting • Limitations • FAQ	• System administrators of IBM Spectrum Scale systems • Application programmers who are experienced with IBM Spectrum Scale systems and familiar with the terminology and concepts in the XDSM standard

Table 1. IBM Spectrum Scale library information units (continued)		
Information unit	**Type of information**	**Intended users**
IBM Spectrum Scale Erasure Code Edition Guide	IBM Spectrum Scale Erasure Code Edition • Summary of changes • Introduction to IBM Spectrum Scale Erasure Code Edition • Planning for IBM Spectrum Scale Erasure Code Edition • Installing IBM Spectrum Scale Erasure Code Edition • Uninstalling IBM Spectrum Scale Erasure Code Edition • Incorporating IBM Spectrum Scale Erasure Code Edition in an Elastic Storage Server (ESS) cluster • Creating an IBM Spectrum Scale Erasure Code Edition storage environment • Using IBM Spectrum Scale Erasure Code Edition for data mirroring and replication • Upgrading IBM Spectrum Scale Erasure Code Edition • Administering IBM Spectrum Scale Erasure Code Edition • Troubleshooting • IBM Spectrum Scale RAID Administration	• System administrators of IBM Spectrum Scale systems • Application programmers who are experienced with IBM Spectrum Scale systems and familiar with the terminology and concepts in the XDSM standard

Table 1. IBM Spectrum Scale library information units (continued)

Information unit	Type of information	Intended users
IBM Spectrum Scale Container Storage Interface Driver Guide	This guide provides the following information: • Summary of changes • Introduction • Planning • Installation • Upgrading • Configurations • Using IBM Spectrum Scale Container Storage Interface Driver • Managing IBM Spectrum Scale when used with IBM Spectrum Scale Container Storage Interface driver • Cleanup • Limitations • Troubleshooting	• System administrators of IBM Spectrum Scale systems • Application programmers who are experienced with IBM Spectrum Scale systems and familiar with the terminology and concepts in the XDSM standard
IBM Spectrum Scale on AWS Guide	This guide provides the following information: • Summary of changes • Introduction to IBM Spectrum Scale on AWS • Setting up the IBM Spectrum Scale environment in the AWS Cloud • Deploying IBM Spectrum Scale on AWS • Creating custom AMI • Cluster lifecycle management • Accessing IBM Spectrum Scale GUI in AWS • Active file management on AWS • Upgrading IBM Spectrum Scale • Cleaning up the cluster and the stack • Data security and AWS Identity and Access Management • Diagnosing and cleaning-up deployment failures • Collecting debug data • Troubleshooting • Frequently Asked Questions	• System administrators of IBM Spectrum Scale systems • Application programmers who are experienced with IBM Spectrum Scale systems and familiar with the terminology and concepts in the XDSM standard

Prerequisite and related information

For updates to this information, see IBM Spectrum Scale in IBM Documentation.

For the latest support information, see the IBM Spectrum Scale FAQ in IBM Documentation.

Conventions used in this information

Table 2 on page xxxix describes the typographic conventions used in this information. UNIX file name conventions are used throughout this information.

Note: Users of IBM Spectrum Scale for Windows must be aware that on Windows, UNIX-style file names need to be converted appropriately. For example, the GPFS cluster configuration data is stored in the `/var/mmfs/gen/mmsdrfs` file. On Windows, the UNIX namespace starts under the `%SystemDrive%\cygwin64` directory, so the GPFS cluster configuration data is stored in the `C:\cygwin64\var\mmfs\gen\mmsdrfs` file.

Table 2. Conventions

Convention	Usage
bold	Bold words or characters represent system elements that you must use literally, such as commands, flags, values, and selected menu options. Depending on the context, **bold** typeface sometimes represents path names, directories, or file names.
bold underlined	bold underlined keywords are defaults. These take effect if you do not specify a different keyword.
constant width	Examples and information that the system displays appear in `constant-width` typeface. Depending on the context, `constant-width` typeface sometimes represents path names, directories, or file names.
italic	*Italic* words or characters represent variable values that you must supply. *Italics* are also used for information unit titles, for the first use of a glossary term, and for general emphasis in text.
<*key*>	Angle brackets (less-than and greater-than) enclose the name of a key on the keyboard. For example, <Enter> refers to the key on your terminal or workstation that is labeled with the word *Enter*.

****	In command examples, a backslash indicates that the command or coding example continues on the next line. For example:
	`mkcondition -r IBM.FileSystem -e "PercentTotUsed > 90" \`
	`-E "PercentTotUsed < 85" -m p "FileSystem space used"`

{*item*}	Braces enclose a list from which you must choose an item in format and syntax descriptions.
[*item*]	Brackets enclose optional items in format and syntax descriptions.
<Ctrl-*x*>	The notation <Ctrl-*x*> indicates a control character sequence. For example, <Ctrl-c> means that you hold down the control key while pressing <c>.
***item*...**	Ellipses indicate that you can repeat the preceding item one or more times.

Table 2. Conventions (continued)

Convention	Usage
|	In *synopsis* statements, vertical lines separate a list of choices. In other words, a vertical line means *Or*. In the left margin of the document, vertical lines indicate technical changes to the information.

Note: CLI options that accept a list of option values delimit with a comma and no space between values. As an example, to display the state on three nodes use `mmgetstate -N` *NodeA*,*NodeB*,*NodeC*. Exceptions to this syntax are listed specifically within the command.

How to send your comments

Your feedback is important in helping us to produce accurate, high-quality information. If you have any comments about this information or any other IBM Spectrum Scale documentation, send your comments to the following e-mail address:

`mhvrcfs@us.ibm.com`

Include the publication title and order number, and, if applicable, the specific location of the information about which you have comments (for example, a page number or a table number).

To contact the IBM Spectrum Scale development organization, send your comments to the following e-mail address:

`scale@us.ibm.com`

Summary of changes

This topic summarizes changes to the IBM Spectrum Scale licensed program and the IBM Spectrum Scale library. Within each information unit in the library, a vertical line (**|**) to the left of text and illustrations indicates technical changes or additions that are made to the previous edition of the information.

Summary of changes
for IBM Spectrum Scale 5.1.1
as updated, July 2021

This release of the IBM Spectrum Scale licensed program and the IBM Spectrum Scale library includes the following improvements. All improvements are available after an upgrade, unless otherwise specified.

- Commands, data types, and programming APIs
- Messages
- Stabilized items
- Deprecated items
- Documentation changes

Important: IBM Spectrum Scale 5.1.x supports Python 3.6 or later. It is strongly recommended that Python 3.6 is installed through the OS package manager (For example, **`yum install python3`**). If you install Python 3.6 by other means, unexpected results might occur, such as failure to install `gpfs.base` for prerequisite checks, and workarounds might be required.

AFM and AFM DR-related changes

AFM-DR filesets support `Immutability` and `appendOnly` flags and all integrated archive manager (IAM) modes. For more information, see *Immutability and appendOnly for AFM-DR filesets* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide* and *Enabling integrated archive manager (IAM) modes on AFM-DR filesets* in the *IBM Spectrum Scale: Administration Guide*.

AFM to Cloud Object Storage

- Supports partial file or object caching. For more information, see *Partial file or object caching for AFM to cloud object storage* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.
- Supports symbolic links (symlinks). For more information, see *Symbolic links* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Big data and analytics changes

For information on changes in IBM Spectrum Scale Big Data and Analytics support and HDFS protocol, see Big Data and Analytics - summary of changes.

IBM Spectrum Scale Container Storage Interface driver changes

For information on changes in the IBM Spectrum Scale Container Storage Interface driver, see IBM Spectrum Scale Container Storage Interface driver - Summary of changes.

IBM Spectrum Scale Erasure Code Edition changes

For information on changes in the IBM Spectrum Scale Erasure Code Edition, see IBM Spectrum Scale Erasure Code Edition - Summary of changes.

File audit logging changes

File audit logging no longer makes use of Kafka message queues. For more information, see *Requirements, limitations, and support for file audit logging* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Generate ACCESS_DENIED events for denial of restricted changes to immutable or appendOnly files.

File system core improvements

Designate a node as performance monitoring collector when you add a node in the cluster
You can designate a node as a performance monitoring node when you add a node by using the **mmaddnode** command. You can do it by using **perfmon** as the **NodeDesignation**. For more information about the new node designation, see *mmaddnode command* in *IBM Spectrum Scale: Command and Programming Reference*.

The **perfmon** designation is ignored if the node is not a Linux node or if the performance monitoring is not configured yet. For more details about how to generate the configuration of the performance monitoring tool, see *mmperfmon command* in *IBM Spectrum Scale: Command and Programming Reference*.

MMBACKUP_SERVER_FROM_OPT option to specify the IBM Spectrum Protect server to be used for backups
Added **MMBACKUP_SERVER_FROM_OPT** option in the **mmbackup** command to specify the IBM Spectrum Protect server to be used when running the **mmbackup** command without the **--tsm-servers** option. For more information about the newly added **MMBACKUP_SERVER_FROM_OPT** parameter, see *mmbackup command* in *IBM Spectrum Scale: Command and Programming Reference*.

df command reports inode capacity and inode usage if quota is disabled
Change in the behavior of **df** command if quota is disabled and **filesetdf** parameter is enabled. If quota is disabled and **filesetdf** is enabled in IBM Spectrum Scale 5.1.1 or later with file system version 5.1.1 or later, then the **df** command reports inode space capacity and inode usage at the independent fileset-level. However, the **df** command reports the block space at the file system-level because the block space is shared with the whole file system.

For more information about the change in the **df** command behavior, see *mmcrfs command* in *IBM Spectrum Scale: Command and Programming Reference*

Configure multiple connections over TCP to optimize the bandwidth usage
Starting with IBM Spectrum Scale 5.1.1, you can configure multiple connections over TCP that enhances the IBM Spectrum Scale communication subsystem to establish multiple TCP connections for communication between each pair of daemons. Using more connections may improve performance and resiliency, particularly for some bonded configurations, which can use more than one physical interface per node with multiple TCP connections.

The number of connections is controlled through the **maxTcpConnsPerNodeConn** parameter, which can be changed by using the **mmchconfig** command. For more information about how to configure the **maxTcpConnsPerNodeConn** parameter, see *mmchconfig command* in *IBM Spectrum Scale: Command and Programming Reference*.

The value that is assigned to **maxTcpConnsPerNodeConn** must be defined after considering certain aspects. For more information about selecting appropriate value for the **maxTcpConnsPerNodeConn** parameter, see *Recommendations for tuning maxTcpConnsPerNodeConn parameter* in *IBM Spectrum Scale: Administration Guide*.

Characters allowed in the keystore password (mmkeyserv command)
Only the following characters are allowed in the keystore password for the **mmkeyserv** command:

0 to 9
A to Z
a to z
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~

For more information, see *mmkeyserv command* in *IBM Spectrum Scale: Command and Programming Reference*.

Policy rules: New file attribute that can be used in the SQL expressions
Introduced the value **d** for the file attribute **MISC_ATTRIBUTE**. This new attribute can be used in the SQL expressions to identify the files whose data is completely contained in the inodes. For example, if the file size is 3 KB, which is less than the default inode size of 4 KB, then the file can

be completely contained in the inode. For more information, see *File attributes in SQL expressions* in *IBM Spectrum Scale: Command and Programming Reference*.

Introduced warning for the users when they try to delete a node that is configured as a performance monitoring collector

If the node is configured as a performance monitoring collector, you need to remove the node from the performance monitoring configuration by using the **mmperfmon config update --collectors** command before you delete the node. Deleting a collector node causes loss of all the collected perfmon stats data on the collector node. For more information about the restrictions for deleting a node, see *Deleting nodes from a GPFS cluster*.

New parameter to specify whether to authenticate notify RPCs related to deadlock detection and amelioration, node overload and expel

Introduced the **sdrNotifyAuthEnabled** parameter in the **mmchconfig** command to specify whether to authenticate the notify RPCs related to deadlock detection and amelioration, node overload, and expel. The scope of this parameter is a single cluster. However, remote clusters may be impacted by this setting, as notify RPCs can be used between remote clusters. For more information about how to set value for the **sdrNotifyAuthEnabled** parameter, see *mmchconfig command* in *IBM Spectrum Scale: Command and Programming Reference*.

Using the --pit-continues-on-error option to process as many as possible files even if errors are encountered while running certain commands

The new `--pit-continues-on-error` option enables the following commands to continue running even if an error occurs during the parallel inode traverse (PIT) phase of the command execution:

- **mmrestripefs**
- **mmchdisk**
- **mmdefragfs**
- **mmdeldisk**
- **mmdelfileset**
- **mmdelsnapshot**

You need to enable the newly added **--pit-continues-on-error** option in these commands to enable this feature. Without enabling this feature, the daemon stops the execution of the command when an error occurs in the PIT phase. Refer to the man pages of these commands in the *IBM Spectrum Scale: Command and Programming Reference* for more details about the **--pit-continues-on-error** option.

--fast option in the mmlssnapshot command

When you specify the **--fast** option, the **mmlssnapshot** command uses a faster way of calculating the amount of storage that is used by a snapshot, minimizing the impact on the performance of the system. This is achieved by getting the required details from the inode of the snapshots without locking the snapshot files, and also by reducing the scanning on the indirect block tree of snapshot files. For more information, see *mmlssnapshot command* in *IBM Spectrum Scale: Command and Programming Reference*.

--on-error-continue option in the mmaddnode and mmdelnode commands

When you add or delete a list of nodes from the cluster, the process fails if it encounters with any errors in one of the nodes. If you select the new **--on-error-continue** option in the **mmaddnode** and **mmdelnode** commands, the daemon continues the process even if there are some errors with some of the nodes. For more information about the **--on-error-continue** option, see *mmaddnode command* and *mmdelnode command* topics in *IBM Spectrum Scale: Command and Programming Reference*.

Support for multiple IP aliases per RoCE RDMA adapter port

IBM Spectrum Scale now supports multiple IP aliases per RoCE RDMA adapter port. The IP aliases help the system to have multiple IP addresses per network interface that are part of internal and external networks. IBM Spectrum Scale automatically selects the correct source and destination

IP address when establishing a connection between two RDMA adapter ports, even if multiple IP addresses are used by the adapter ports.

Support for GPUDirect Storage
IBM Spectrum Scale 5.1.1 introduces GPUDirect Storage as a Tech Preview feature. GPUDirect Storage can improve the performance of GPU-enabled applications by transferring file data directly between IBM Spectrum Scale storage servers and the client GPU. This enhanced data path reduces client CPU utilization incurred by the file system and bypasses potential performance bottlenecks that manifest when file data is staged through intermediate buffers in client system memory. Refer to the following website for additional documentation, restrictions, and requirements: https://www.ibm.com/support/pages/node/6444075

Maximum size of a local read-only cache device
The maximum size of a local read-only cache device is 4 TB. For more information about local read-only cache, see *Local read-only cache* in *IBM Spectrum Scale: Administration Guide*.

LROC: Warnings generated if amount of buffer descriptors is low for LROC device capacity
To reference data stored in LROC, IBM Spectrum Scale might require more buffer descriptors than the default value of buffer descriptors on a particular node. Warnings are written to the `mmfs.log` if the amount of buffer descriptors that are allocated is deemed too low based on the LROC device capacity. This warning would be displayed in System Health as a TIPS event if System Health is enabled. For more information, see *Local read-only cache* in *IBM Spectrum Scale: Administration Guide*.

Monitoring local read-only cache
The health and events of LROC devices can be monitored using **`mmhealth`** command. The LROC component and I/O history can be monitored using **`mmdiag`** command. The file contents in LROC can be monitored using **`mmcachectl`** command. For more information about local read-only cache, see *Monitoring local read-only cache* in *IBM Spectrum Scale: Problem Determination Guide*.

Monitoring encryption
The health and events of the on-disc encryption can be monitored using the **`mmhealth`** command. For more information about encryption, see the *mmhealth command* section in the *IBM Spectrum Scale: Command and Programming Reference* guide and the *Encryption events* section in the *IBM Spectrum Scale: Problem Determination Guide*.

Installation toolkit changes

- Migration to the Ansible automation platform:
  - Enables scaling up to more number of nodes
  - Avoids issues that arise due to using an agent-based tooling infrastructure such as Chef
  - Enables parity with widely-adopted, modern tooling infrastructure
- Support for Red Hat Enterprise Linux 8.4 on x86_64, PPC64LE, and s390x
- Support for Ubuntu 20.04 on PPC64LE
- Support for multiple recovery groups in IBM Spectrum Scale Erasure Code Edition
- Simplification of file system creation:

  The file system is created during the installation phase rather than the deployment phase.
- Support for IPv6 addresses
- Support for the CES interface mode
- IBM Spectrum Scale deployment playbooks are now open sourced on GitHub. Users can access the playbooks from the external GitHub repository and implement in their environment on their own. For more information, see https://github.com/IBM/ibm-spectrum-scale-install-infra.

  Note: The installation toolkit (**`./spectrumscale`** command) is only available as part of the IBM Spectrum Scale installation packages that can be downloaded from IBM FixCentral.
- Added an option to enable workload prompt to allow administrators to stop and migrate workloads before a node is shut down for upgrade.

For more information, see *Overview of the installation toolkit* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Management API changes

Added the following endpoints:

- GET Access/acls
- GET Access/acls/{userGroup}
- GET Diagnostic/snap
- GET Diagnostic/snap/{snapPath}
- GET Filesystems/{filesystemName}/pools/{poolName}
- GET Filesystems/{filesystemName}/pools
- GET Nodes/health/config/webhook/listEventWebhook
- GET nodes/network
- GET scalemgmt/v2/gnr/diskmgmt/vdiskset/server/list/{nodeClass}
- GET scalemgmt/v2/encryption/rkmClients
- GET scalemgmt/v2/encryption/rkmKeys
- GET scalemgmt/v2/encryption/rkmServer
- GET scalemgmt/v2/encryption/rkms
- GET scalemgmt/v2/encryption/rkmTenants
- POST Access/acls/{userGroup}
- POST Diagnostic/snap
- POST scalemgmt/v2/encryption/serverAdd
- POST /scalemgmt/v2/encryption/addTenant
- POST /scalemgmt/v2/encryption/createKey
- POST /scalemgmt/v2/encryption/clients
- POST /scalemgmt/v2/encryption/registerClient
- POST Nodes/health/config/webhook/addEventWebhook
- PUT /scalemgmt/v2/encryption/deregisterClient
- PUT scalemgmt/v2/encryption/updateClientkeys
- PUT scalemgmt/v2/encryption/serverUpdate
- PUT filesystems/{filesystemName}/suspend
- PUT filesystems/{filesystemName}/resume
- PUT health/config/{interval}
- PUT filesystem/{filesystemName}/maintenanceMode/{maintenanceModeoption}
- PUT Access/acls/{userGroup}
- PUT Diagnostic/snap/{snapPath}/pmr/{pmrID}
- DELETE Access/acls/{userGroup}
- DELETE Access/acls/{userGroup}/entry/{entryId}
- DELETE Diagnostic/snap/{snapPath}
- DELETE Nodes/health/config/webhook/deleteEventWebhook
- DELETE /scalemgmt/v2/encryption/clientName/{clientName}
- DELETE /scalemgmt/v2/encryption/deleteTenant
- DELETE /scalemgmt/v2/encryption/serverName/{serverName}

The following endpoint is modified:

- POST filesystems/{filesystemName}/filesets/{filesetName}/afmctl
- POST Filesystems/{filesystemName}/directory/{path}
- PUT filesystems/filesystemName/owner/path

Management GUI changes

- Ports allocated to performance monitoring service are changed to 9980 for IBM Spectrum Scale and 9981 for IBM Spectrum Scale Container Native Storage Access (CNSA) .
- Multi-Factor Authentication is enabled for GUI users. For more information, see the topic *Configuring multi-factor authentication for GUI users* in the *IBM Spectrum Scale: Administration Guide*
- Configuration parameters are provided under **Services** > **GPFS daemon** to define the manner in which the available limit or quota of data storage and file system sizes are displayed.
- User passwords can now be configured so that they never expire. For more information, see the topic *Create GUI users and assign user permissions* in the *IBM Spectrum Scale: Administration Guide*
- The number of API requests that can be queued in one second is defined under **Services** > **GUI** to eliminate the possibility of a Denial of Service (DOS) state owing to server resources being exhausted.
- The **Services** > **GUI** provides options to enable the users with Security Administrator privileges to define the number of failed login attempts and the maximum duration after which the user account is locked.
- Access Control lists (ACL) are used to define access to individual REST APIs at the user group level. For more information, see the topic *IBM Spectrum Scale management API* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*
- The session timeout is enabled by default and predefined at 30 minutes after which the user is automatically logged out. The **Services** > **GUI** provides the option to configure this parameter and set a new session timeout duration as required.
- Last login details displayed for users provides for an additional security insight.
- Deletion schedules are defined under the **Files** > **Snapshots** section.
- The **Resolve Event** option issues the **`mmhealth event resolve`** command to fix events. For more information, see the topic *Monitoring events using GUI* in the *IBM Spectrum Scale: Administration Guide*.
- A backup of Truststore files must be created prior to all GUI upgrades. For more information, see the topic *Configuring external authentication for GUI users* in the *IBM Spectrum Scale: Administration Guide*
- The recursive search option enables the mapping of nested groups while authenticating GUI users. For more information, see the topic *Configuring external authentication for GUI users* in the *IBM Spectrum Scale: Administration Guide*.
- Installing *iptables* is an important prerequisite for GUI installation on Linux based operating systems. For more information, see the topic *Manually installing IBM Spectrum Scale management GUI* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.
- In Dual stack configuration, pmcollectors must have the same IP protocol versions defined within the cluster. To establish a successful connection with the management GUI, the name resolution must be manually defined in the ZIMonCollector.cfg file. For more information, see the topic *Configuring the collector* in the *IBM Spectrum Scale: Problem Determination Guide*.

NFS changes

Support for z/OS NFS clients with or without Kerberos. For more information about the supported OS and NFS clients, see Supported IBM Spectrum Scale FAQ in IBM Documentation.

Packaging changes

- The `gpfs.protocols-support` package is removed from the IBM Spectrum Scale installation images for Linux.

Python-related changes

From IBM Spectrum Scale release 5.1.0, all Python code in the IBM Spectrum Scale product is converted to Python 3. The minimum supported Python version is 3.6.

For compatibility reasons on IBM Spectrum Scale 5.1.0.x and later on Red Hat Enterprise Linux 7.x (7.7 and later), a few Python 2 files are packaged and they might trigger dependency-related messages. In certain scenarios, Python 2.7 might also be required to be installed. Multiple versions of Python can co-exist on the same system. For more information, see the entry about **mmadquery** in *Guidance for Red Hat Enterprise Linux 8.x on IBM Spectrum Scale nodes* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

The Python code in IBM Spectrum Scale 5.0.y or earlier continues to be in Python 2.

Tip:

- IBM Spectrum Scale 5.1.x.x uses Python 3 code and it runs best with operating systems that also use Python 3 internally such as Red Hat Enterprise Linux 8.x, SLES 15, and Ubuntu 20.04.
- IBM Spectrum Scale 5.0.x.x uses Python 2 code and it runs best with operating systems that also use Python 2 internally such as Red Hat Enterprise Linux 7.x.

System health changes

- New sub-commands added to the **mmhealth node** command

  The **mmhealth config monitor** command displays the health monitoring service status of a node. For more information, see the *mmhealth* command in the *IBM Spectrum Scale: Command and Programming Reference*.

- New sub-commands to the **mmperfmon config** command

  New API sub-command added to the **mmperfmon config** command which can be used to add, update, show, and delete API keys. For more information, see the *mmperfmon* command in the *IBM Spectrum Scale: Command and Programming Reference* and the *Configuring performance monitoring API keys* in the *IBM Spectrum Scale: Problem Determination Guide*.

- Local Cache and Encryption component added to the **mmhealth node show** and **mmhealth cluster show** commands.

  For more information, see the *mmhealth command* section in the *IBM Spectrum Scale: Command and Programming Reference* guide and the *Local cache events* and *Encryption events* section in the *IBM Spectrum Scale: Problem Determination Guide*.

- Changes to the performance monitoring tool.

  Starting with IBM Spectrum Scale version 5.1.1, a new Linux operating system user, `scalepm`, is added to the host. This user id is used by the pmcollector to run the process in the context of the new user. For more information, see the *Using the performance monitoring tool* section in the *IBM Spectrum Scale: Problem Determination Guide*.

- Updates to firewall recommendations for the Performance Monitoring tool.

  New ports have been added, while some of the old ports are no longer required by the Performance Monitoring tool. For more information, see the *Firewall recommendations for Performance Monitoring tool* section in the *IBM Spectrum Scale: Administration Guide*.

- New events added for the following:
  - File system events
  - Canister events

  For more information, see the *Events* section in the *IBM Spectrum Scale: Problem Determination Guide*.

Call home changes

Call home has the following improvements:

- New commands are added to scheduled data upload list. For more information, see the *Scheduled data upload* section in the *IBM Spectrum Scale: Problem Determination Guide*.

Linux on Z changes

IBM Spectrum Scale supports thin provisioning of volumes with certain IBM Spectrum Virtualize hardware configurations like V7000. For more information, see *What are the considerations for using block storage systems that support thinly provisioned volumes (thin provisioning)?* in IBM Spectrum Scale FAQ in IBM Documentation

IBM Spectrum Scale on AWS changes

For information on changes in IBM Spectrum Scale on AWS 1.3.1.1, see Summary of changes.

Commands, data types, and programming APIs

The following section lists the modifications to the documented commands, structures, and subroutines:

New commands

There are no new commands.

New structures

There are no new structures.

New subroutines

There are no new subroutines.

New user exits

There are no new user exits.

Changed commands

- **mmaddcallback**
- **mmaddnode**
- **mmadquery**
- **mmapplypolicy**
- **mmaudit**
- **mmauth**
- **mmbackup**
- **mmcachectl**
- **mmcallhome**
- **mmces**
- **mmchattr**
- **mmchcluster**
- **mmchconfig**
- **mmchdisk**
- **mmcheckquota**
- **mmchfs**
- **mmchlicense**
- **mmchmgr**
- **mmchnode**
- **mmchnsd**
- **mmchqos**
- **mmcloudgateway**
- **mmcrcluster**
- **mmcrfileset**
- **mmcrfs**
- **mmcrsnapshot**

- **mmdefquotaon**
- **mmdefragfs**
- **mmdeldisk**
- **mmdelnode**
- **mmdelfileset**
- **mmdiag**
- **mmhealth**
- **mmimportfs**
- **mmkeyserv**
- **mmlsdisk**
- **mmlsfileset**
- **mmlsfs**
- **mmlslicense**
- **mmlssnapshot**
- **mmnetverify**
- **mmobj**
- **mmpmon**
- **mmperfmon**
- **mmprotocoltrace**
- **mmqos**
- **mmrestripefile**
- **mmrestripefs**
- **mmrpldisk**
- **mmsdrrestore**
- **spectrumscale**
- **mmsmb**
- **mmwatch**

Changed structures
There are no changed structures.

Changed subroutines
There are no changed subroutines.

Deleted commands
There are no deleted commands.

Deleted structures
There are no deleted structures.

Deleted subroutines
There are no deleted subroutines.

Messages
The following are the new, changed, and deleted messages:

New messages
6027-1303, 6027-1308, 6027-2051, 6027-2052, 6027-3109, 6027-3110, 6027-3111, 6027-3112, 6027-3113, 6027-3114, 6027-3264, 6027-3333, 6027-3334, 6027-3515, 6027-3614, 6027-4211, and 6027-4302.

Changed messages
6027-1752, 6027-2414, 6027-2415, and 6027-2416.

Deleted messages

None.

Stabilized, deprecated, and discontinued features in 5.1.1

A feature is a stabilized feature if there is no plan to deprecate or remove this capability in a subsequent release of the product. It remains supported and current with updates to the operating systems. You do not need to change any of your existing applications and scripts that use a stabilized function now. You should not expect significant new functionality or enhancements to these features.

Table 3. Features stabilized in IBM Spectrum Scale 5.1.1

Category	Stabilized Functionality	Recommended Action
FPO	All	FPO and SNC remain available. However, it is recommended to limit the size of deployments to 32 nodes. There are no plans for significant new functionality in FPO nor increases in scalability. The strategic direction for storage using internal drives and storage rich servers is IBM Spectrum Scale Erasure Code Edition (ECE).
HDFS Transparency	HDP support, HDFS Transparency 3.1.0 or earlier	Plan to migrate to Cloudera supported releases.

A feature is a *deprecated* feature if that specific feature is supported in the current release, but the support might be removed in a future release. In some cases, it might be advisable to plan to discontinue the use of deprecated functionality.

Table 4. Features deprecated in IBM Spectrum Scale 5.1.1

Category	Deprecated functionality	Recommended Action
Platforms	AIX support for IBM Power7 systems	Plan to migrate to newer generations of Power systems.
Message queue	Kafka message queue for clustered watch folders and file audit logging	The Kafka message queue is no longer installed as a part of the IBM Spectrum Scale installation. If you are using Kafka for other purposes, install it separately from IBM Spectrum Scale. If you are upgrading to version 5.1.1, you must disable file audit logging and clustered watch folder and then run **mmmsgqueue config --remove** before the upgrade, or upgrade to 5.1.0.x, run **mmmsgqueue config --remove-msgqueue**, and then proceed with the upgrade to 5.1.1.

Table 4. Features deprecated in IBM Spectrum Scale 5.1.1 (continued)		
Category	**Deprecated functionality**	**Recommended Action**
Security	Support for Vormetric DSM V5	Upgrade to Vormetric DSM V6.2 or later. For more information, see the topic *Preparation for encryption* in the *IBM Spectrum Scale: Administration Guide*.
Protocols	**mmcesdr** command (Protocols cluster disaster recovery)	Use AFM and AFM DR to set up your own replication strategies between clusters.
HDFS Transparency	HDFS Transparency 2.7 or earlier	Plan to migrate to Cloudera supported releases.

A feature is a *Discontinued* feature if it has been removed in a release and is no longer available. You need to make changes if you were using that functionality in previous releases.

Table 5. Features discontinued in IBM Spectrum Scale 5.1.1		
Category	**Discontinued functionality**	**Recommended Action**
Security	The use of TLS 1.0 and 1.1 for authorization within and between IBM Spectrum Scale clusters.	Upgrade to TLS 1.2 or later.
GUI/REST API	The use of TLS 1.0 and 1.1 for authorization with the GUI/ REST API server	Upgrade to TLS 1.2 or later.
Platforms	Encryption acceleration library for Power7 (CLIC)	If encryption performance is critical, migrate to newer generations of Power Systems.
	Big Endian Power servers	Upgrade to newer generations of Power systems or remain on IBM Spectrum Scale 5.0.5.
	Linux support for IBM Power7 systems	Plan to migrate to newer generations of Power systems.
Protocols	iSCSI as a target for remote boot	Use some other block services provider.
Containers	Storage Enabler for Containers (SEC)	Migrate to Container Storage Interface (CSI).
Cluster configuration	The primary and secondary configuration server functionality. Instead of this, clusters must use CCR.	The default configuration service is CCR, and new clusters are created using CCR. Use the **mmlscluster** command to determine the configuration service. If not operating with CCR, change to that mode with the **mmchcluster --ccr-enable** command.

Table 5. Features discontinued in IBM Spectrum Scale 5.1.1 (continued)		
Category	**Discontinued functionality**	**Recommended Action**
Installation toolkit	NTP configuration	Do time synchronization configuration manually across all nodes in a cluster by using the available method.
	File and object authentication configuration	Use the **mmuserauth** command to do file and object authentication configuration.
	NSD balance	Use the **./spectrumscale nsd servers** command to balance the NSD preferred node between the primary and secondary nodes. For example: `./spectrumscale nsd servers setprimary -s node1.example.com nsd1`

Documentation changes

Product guides in EPUB format

Product guides in EPUB format are no longer shipped with the documentation.

List of documentation changes in product guides and respective IBM Documentation sections

The following is a list of documentation changes including changes in topic titles, changes in placement of topics, and deleted topics:

Table 6. List of changes in documentation		
Guide	**IBM Documentation section**	**List of changes**
Problem Determination Guide	Troubleshooting	Removed the following topics: • *Installation toolkit hangs indefinitely during a GPFS state check* • *Installation toolkit hangs during a subsequent session after the first session was terminated* • *Package conflict on SLES nodes while doing installation, deployment, or upgrade using installation toolkit* • *Chef crashes during installation, upgrade, or deployment using the installation toolkit* • *Chef commands require configuration changes to work in an environment that requires proxy servers* • *Chef client-related issue during upgrade precheck on Ubuntu 20.04 nodes* • *Upgrading Ubuntu causes Chef client to be upgraded to an unsupported version for the installation toolkit*

Table 6. List of changes in documentation (continued)		
Guide	**IBM Documentation section**	**List of changes**
Administration Guide	Configuring	Updated the following topics: • *Configuring IBM Spectrum Scale cluster to enable Cinder driver with RHOSP* • *Deploying IBM Spectrum Scale Cinder backend configuration in RHOSP* • *Sample IBM Spectrum Scale Cinder configuration YAML file* • *Actions that the mmwatch all upgrade command takes*

Chapter 1. Command reference

A list of all the GPFS commands and a short description of each is presented in this topic.

Table 7 on page 1 summarizes the GPFS-specific commands.

Table 7. GPFS commands

Command	Purpose
“gpfs.snap command” on page 8	Creates an informational system snapshot at a single point in time. This system snapshot consists of information such as cluster configuration, disk configuration, network configuration, network status, GPFS logs, dumps, and traces.
“mmaddcallback command” on page 12	Registers a user-defined command that GPFS will execute when certain events occur.
“mmadddisk command” on page 28	Adds disks to a GPFS file system.
“mmaddnode command” on page 34	Adds nodes to a GPFS cluster.
“mmadquery command” on page 39	Queries and validates Active Directory (AD) server settings.
“mmafmconfig command” on page 45	Can be used to manage home caching behavior and mapping of gateways and home NFS exported servers.
“mmafmcosaccess command” on page 48	This command is used to map a directory in an AFM to could object storage fileset to a bucket on a cloud object storage.
“mmafmcosconfig command” on page 50	This command used to create and display an AFM to cloud object storage fileset.
“mmafmcosctl command” on page 55	This command is used to control the `download` and `upload` operations between an AFM to cloud object storage fileset and a cloud object storage.
“mmafmcoskeys command” on page 58	This command is used to manage access and secret keys of a bucket on a cloud object storage.
“mmafmctl command” on page 61	This command is for various operations and reporting information on all filesets. It is recommended to read the *IBM Spectrum Scale: Administration Guide* AFM and AFM Disaster Recovery chapters along with this manual for detailed description of the functions.
“mmafmlocal command” on page 78	Provides a list of cached files and file statistics such as inode number, allocated blocks, and so on.
“mmapplypolicy command” on page 80	Deletes files, migrates files between storage pools, or does file compression or decompression in a file system as directed by policy rules.
“mmaudit command” on page 92	Manages setting and viewing the file audit logging configuration in IBM Spectrum Scale.
“mmauth command” on page 96	Manages secure access to GPFS file systems.
“mmbackup command” on page 101	Performs a backup of a GPFS file system or independent fileset to an IBM Spectrum Protect server.

Table 7. GPFS commands (continued)	
Command	**Purpose**
“mmbackupconfig command” on page 111	Collects GPFS file system configuration information.
“mmbuildgpl command” on page 113	Manages prerequisite packages for Linux and builds the GPFS portability layer.
“mmcachectl command” on page 115	Displays information about files and directories in the local page pool cache.
“mmcallhome command” on page 118	Manages the call home operations.
“mmces command” on page 133	Manages CES configuration.
“mmcesdr command” on page 148	Manages protocol cluster disaster recovery.
“mmchattr command” on page 157	Changes attributes of one or more GPFS files.
“mmchcluster command” on page 165	Changes GPFS cluster configuration data.
“mmchconfig command” on page 170	Changes GPFS configuration parameters.
“mmchdisk command” on page 212	Changes state or parameters of one or more disks in a GPFS file system.
“mmcheckquota command” on page 220	Checks file system user, group and fileset quotas.
“mmchfileset command” on page 224	Changes the attributes of a GPFS fileset.
“mmchfs command” on page 232	Changes the attributes of a GPFS file system.
“mmchlicense command” on page 239	Controls the type of GPFS license associated with the nodes in the cluster.
“mmchmgr command” on page 242	Assigns a new file system manager node or cluster manager node.
“mmchnode command” on page 244	Changes node attributes.
“mmchnodeclass command” on page 251	Changes user-defined node classes.
“mmchnsd command” on page 254	Changes Network Shared Disk (NSD) configuration attributes.
“mmchpolicy command” on page 258	Establishes policy rules for a GPFS file system.
“mmchpool command” on page 261	Modifies storage pool properties.
“mmchqos command” on page 263	Controls Quality of Service for I/O operations (QoS) settings for a file system.
“mmclidecode command” on page 272	Decodes the parseable command output field.
“mmclone command” on page 274	Creates and manages file clones.
“mmcloudgateway command” on page 277	Creates and manages the cloud storage tier.
“mmcrcluster command” on page 306	Creates a GPFS cluster from a set of nodes.
“mmcrfileset command” on page 311	Creates a GPFS fileset.
“mmcrfs command” on page 318	Creates a GPFS file system.
“mmcrnodeclass command” on page 333	Creates user-defined node classes.
“mmcrnsd command” on page 335	Creates Network Shared Disks (NSDs) used by GPFS.

Table 7. GPFS commands (continued)	
Command	**Purpose**
“mmcrsnapshot command” on page 340	Creates a snapshot of a file system or fileset at a single point in time.
“mmdefedquota command” on page 345	Sets default quota limits.
“mmdefquotaoff command” on page 349	Deactivates default quota limit usage.
“mmdefquotaon command” on page 352	Activates default quota limit usage.
“mmdefragfs command” on page 356	Reduces disk fragmentation by increasing the number of full free blocks available to the file system.
“mmdelacl command” on page 360	Deletes a GPFS access control list.
“mmdelcallback command” on page 362	Deletes one or more user-defined callbacks from the GPFS system.
“mmdeldisk command” on page 364	Deletes disks from a GPFS file system.
“mmdelfileset command” on page 369	Deletes a GPFS fileset.
“mmdelfs command” on page 373	Removes a GPFS file system.
“mmdelnode command” on page 375	Removes one or more nodes from a GPFS cluster.
“mmdelnodeclass command” on page 379	Deletes user-defined node classes.
“mmdelnsd command” on page 381	Deletes Network Shared Disks (NSDs) from the GPFS cluster.
“mmdelsnapshot command” on page 383	Deletes a GPFS snapshot.
“mmdf command” on page 387	Queries available file space on a GPFS file system.
“mmdiag command” on page 391	Displays diagnostic information about the internal GPFS state on the current node.
“mmdsh command” on page 399	Runs commands on multiple nodes or network connected hosts at the same time.
“mmeditacl command” on page 401	Creates or changes a GPFS access control list.
“mmedquota command” on page 404	Sets quota limits.
“mmexportfs command” on page 408	Retrieves the information needed to move a file system to a different cluster.
“mmfsck command” on page 410	Checks and repairs a GPFS file system.
“mmfsctl command” on page 424	Issues a file system control request.
“mmgetacl command” on page 428	Displays the GPFS access control list of a file or directory.
“mmgetstate command” on page 431	Displays the state of the GPFS daemon on one or more nodes.
“mmhealth command” on page 441	Monitors health status of nodes.
“mmhadoopctl command” on page 434	Installs and sets up the GPFS connector for a Hadoop distribution; starts or stops the GPFS connector daemon on a node.
“mmhdfs command” on page 436	The **mmhdfs** command configures and manages the IBM Spectrum Scale HDFS Transparency components.

Table 7. GPFS commands (continued)	
Command	**Purpose**
“mmimgbackup command” on page 458	Performs a backup of a single GPFS file system metadata image.
“mmimgrestore command” on page 462	Restores a single GPFS file system from a metadata image.
“mmimportfs command” on page 465	Imports into the cluster one or more file systems that were created in another GPFS cluster.
“mmkeyserv command” on page 469	Manages encryption key servers and clients.
“mmlinkfileset command” on page 485	Creates a junction that references the root directory of a GPFS fileset.
“mmlsattr command” on page 487	Queries file attributes.
“mmlscallback command” on page 490	Lists callbacks that are currently registered in the GPFS system.
“mmlscluster command” on page 492	Displays the current configuration information for a GPFS cluster.
“mmlsconfig command” on page 495	Displays the current configuration data for a GPFS cluster.
“mmlsdisk command” on page 497	Displays the current configuration and state of the disks in a file system.
“mmlsfileset command” on page 501	Displays attributes and status for GPFS filesets.
“mmlsfs command” on page 506	Displays file system attributes.
“mmlslicense command” on page 511	Displays information about the IBM Spectrum Scale node licensing designation or about disk and cluster capacity.
“mmlsmgr command” on page 515	Displays which node is the file system manager for the specified file systems or which node is the cluster manager.
“mmlsmount command” on page 517	Lists the nodes that have a given GPFS file system mounted.
“mmlsnodeclass command” on page 520	Displays node classes defined in the system.
“mmlsnsd command” on page 522	Displays Network Shared Disk (NSD) information for the GPFS cluster.
“mmlspolicy command” on page 526	Displays policy information.
“mmlspool command” on page 528	Displays information about the known storage pools.
“mmlsquota command” on page 535	Displays quota information for a user, group, or fileset.
“mmlsqos command” on page 530	Displays the I/O performance values of a file system, when you enable Quality of Service for I/O operations (QoS) with the mmchqos command.
“mmlssnapshot command” on page 540	Displays GPFS snapshot information.
“mmmigratefs command” on page 543	Performs needed conversions to support new file system features.

Table 7. GPFS commands (continued)	
Command	**Purpose**
“mmmount command” on page 545	Mounts GPFS file systems on one or more nodes in the cluster.
“mmnetverify command” on page 548	Verifies network configuration and operation in a cluster.
“mmnfs command” on page 560	Manages NFS exports and configuration.
“mmnsddiscover command” on page 571	Rediscovers paths to the specified network shared disks.
“mmobj command” on page 573	Manages configuration of Object protocol service, and administers storage policies for object storage, unified file and object access, and multi-region object deployment.
“mmperfmon command” on page 590	Configures the Performance Monitoring tool and lists the performance metrics.
“mmpmon command” on page 604	Manages performance monitoring and displays performance information.
“mmprotocoltrace command” on page 610	Starts, stops, and monitors tracing for the CES protocols.
“mmpsnap command” on page 614	Creates or deletes identical snapshots on the cache and home clusters, or shows the status of snapshots that have been queued up on the gateway nodes.
“mmputacl command” on page 617	Sets the GPFS access control list for the specified file or directory.
“mmqos command” on page 619	Controls the Quality of Service for I/O operations (QoS) settings for a file system.
“mmquotaoff command” on page 651	Deactivates quota limit checking.
“mmquotaon command” on page 654	Activates quota limit checking.
“mmreclaimspace command” on page 657	Reclaims free space on a GPFS file system.
“mmremotecluster command” on page 660	Manages information about remote GPFS clusters.
“mmremotefs command” on page 663	Manages information needed for mounting remote GPFS file systems.
“mmrepquota command” on page 666	Displays file system user, group, and fileset quotas.
“mmrestoreconfig command” on page 671	Restores file system configuration information.
“mmrestorefs command” on page 675	Restores a file system or an independent fileset from a snapshot.
“mmrestripefile command” on page 678	Rebalances or restores the replication factor of the specified files, performs any incomplete or deferred file compression or decompression, or detects and repairs data and directory block replica mismatches in the specified files.

Table 7. GPFS commands (continued)	
Command	**Purpose**
"mmrestripefs command" on page 682	Rebalances or restores the replication factor of all the files in a file system. Alternatively, this command performs any incomplete or deferred file compression or decompression of all the files in a file system.
"mmrpldisk command" on page 690	Replaces the specified disk.
"mmsdrrestore command" on page 697	Restores the latest GPFS system files on the specified nodes.
"mmsetquota command" on page 702	Sets quota limits.
"mmshutdown command" on page 706	Unmounts all GPFS file systems and stops GPFS on one or more nodes.
"mmsmb command" on page 709	Administers SMB shares, export ACLs, and global configuration.
"mmsnapdir command" on page 722	Controls how the special directories that connect to snapshots appear.
"mmstartup command" on page 726	Starts the GPFS subsystem on one or more nodes.
"mmtracectl command" on page 728	Sets up and enables GPFS tracing.
"mmumount command" on page 732	Unmounts GPFS file systems on one or more nodes in the cluster.
"mmunlinkfileset command" on page 735	Removes the junction to a GPFS fileset.
"mmuserauth command" on page 738	Manages the authentication configuration of file and object access protocols. The configuration allows protocol access methods to authenticate users who need to access data that is stored on the system over these protocols.
"mmwatch command" on page 764	Administers clustered watch folder watches and lists all of the active watch folder API watches.
"mmwinservctl command" on page 770	Manages the `mmwinserv` Windows service.
"spectrumscale command" on page 773	Installs and configures GPFS; adds nodes to a cluster; deploys and configures protocols and performance monitoring tools; configures call home and file audit logging; and upgrades GPFS and protocols.

The following commands are specific to IBM Spectrum Scale RAID and are documented in *IBM Spectrum Scale RAID: Administration*:

- `mmaddcomp`
- `mmaddcompspec`
- `mmaddpdisk`
- `mmchcarrier`
- `mmchcomp`
- `mmchcomploc`
- `mmchenclosure`
- `mmchfirmware`
- `mmchpdisk`

- mmchrecoverygroup
- mmcrrecoverygroup
- mmcrvdisk
- mmdelcomp
- mmdelcomploc
- mmdelcompspec
- mmdelpdisk
- mmdelrecoverygroup
- mmdelvdisk
- mmdiscovercomp
- mmgetdisktopology
- mmlscomp
- mmlscomploc
- mmlscompspec
- mmlsenclosure
- mmlsfirmware
- mmlspdisk
- mmlsrecoverygroup
- mmlsrecoverygroupevents
- mmlsvdisk
- mmsyncdisplayid

gpfs.snap command

Creates an informational system snapshot at a single point in time. This system snapshot consists of information such as cluster configuration, disk configuration, network configuration, network status, GPFS logs, dumps, and traces.

Synopsis

```
gpfs.snap [-d OutputDirectory] [-m | -z]
          [-a | -N {Node[,Node...] | NodeFile | NodeClass}]
          [--check-space | --no-check-space | --check-space-only]
          [--cloud-gateway {NONE |BASIC |FULL} ] [--full-collection] [--deadlock [--quick] |
           --limit-large-files {YYYY:MM:DD:HH:MM | NumberOfDaysBack | latest}]
          [--exclude-aix-disk-attr] [--exclude-aix-lvm] [--exclude-merge-logs]
          [--exclude-net] [--gather-logs] [--mmdf] [--performance] [--prefix]
          [--protocol ProtocolType[,ProtocolType,...]] [--timeout Seconds]
          [--purge-files KeepNumberOfDaysBack][--hadoop]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the **gpfs.snap** command as the main tool to gather data when a GPFS problem is encountered, such as a hung file system, a hung GPFS command, or a daemon assert.

The **gpfs.snap** command gathers information (for example, GPFS internal dumps, traces, and kernel thread dumps) to solve a GPFS problem.

Note: By default, large debug files are now a delta collection, which means that they are only collected when there are new files since the previous run of **gpfs.snap**. To override this default behavior, use either the `--limit-large-files` or `--full-collection` options.

Note: This utility program is a service tool and options might change dynamically. The tool impacts performance and occupies disk space when it runs.

Parameters

-d *OutputDirectory*
Specifies the output directory where the snapshot information is stored. You cannot specify a directory that is located in a GPFS file system that is managed by the same cluster that you are running the `gpfs.snap` command against. The default output directory is `/tmp/gpfs.snapOut`.

-m
Specifying this option is equivalent to specifying `--exclude-merge-logs` with `-N`.

-z
Collects **gpfs.snap** data only from the node on which the command is invoked. No master data is collected.

-a
Directs **gpfs.snap** to collect data from all nodes in the cluster. This value is the default.

-N {*Node*[,*Node ...*] | *NodeFile* | *NodeClass*}
Specifies the nodes from which to collect `gpfs.snap` data. This option supports all defined node classes. For more information about how to specify node names, see *Specifying nodes as input to GPFS commands* in *IBM Spectrum Scale: Administration Guide*.

--check-space
Specifies that space checking is performed before data is collected.

--no-check-space
Specifies that no space checking is performed. This value is the default.

--check-space-only
Specifies that only space checking is performed. No data is collected.

--cloud-gateway {NONE | BASIC | FULL}
When this option is set to NONE, no Transparent cloud tiering data is collected. With the BASIC option, when the Transparent cloud tiering service is enabled, the snap collects information such as logs, traces, Java™ cores, along with minimal system and IBM Spectrum Scale cluster information specific to Transparent cloud tiering. No customer sensitive information is collected.

Note: The default behavior of the `gpfs.snap` command includes basic information of Transparent cloud tiering, in addition to the GPFS information.

With the FULL option, extra details such as Java Heap dump are collected, along with the information captured with the BASIC option.

--full-collection
Specifies that all large debug files are collected instead of the default behavior that collects only new files since the previous run of `gpfs.snap`.

--deadlock
Collects only the minimum amount of data necessary to debug a deadlock problem. Part of the data that is collected is the output of the `mmfsadm dump all` command. This option ignores all other options except for `-a`, `-N`, `-d`, and `--prefix`.

--quick
Collects less data when specified along with the `--deadlock` option. The output includes `mmfsadm dump most`, `mmfsadm dump kthreads`, and 10 seconds of trace in addition to the usual `gpfs.snap` output.

--limit-large-files {*YYYY:MM:DD:HH:MM* | *NumberOfDaysBack* | `latest`}]
Specifies a time limit to reduce the number of large files collected.

--exclude-aix-disk-attr
Specifies that data about AIX disk attributes is not collected. Collecting data about AIX disk attributes on an AIX node that has a large number of disks might be very time-consuming, so using this option might help improve performance.

--exclude-aix-lvm
Specifies that data about the AIX Logical Volume Manager (LVM) is not collected.

--exclude-merge-logs
Specifies that merge logs and waiters is not collected.

--exclude-net
Specifies that network-related information is not collected.

--gather-logs
Gathers, merges, and chronologically sorts all of the `mmfs.log` files. The results are stored in the directory that is specified with `-d` option.

--mmdf
Specifies that `mmdf` output is collected.

--performance
Specifies that performance data is to be gathered. It is recommended to issue the **gpfs.snap** command with **-a** option on all nodes or on the pmcollector node that has an `ACTIVE THRESHOLD MONITOR` role. Starting from IBM Spectrum Scale 5.1.0, the performance data package includes the performance monitoring report from the top metric for last 24 hours as well.

Note: The performance script can take up to 30 minutes to run. Therefore, the script is not included when all other types of protocol information are gathered by default. Specifying this option is the only way to turn on the gathering of performance data.

--prefix
Specifies that the prefix name `gpfs.snap` is added to the tar file.

--protocol *ProtocolType*[,*ProtocolType*,...]
Specifies the type or types of protocol information to be gathered. By default, whenever any protocol is enabled on a file system, information is gathered for all types of protocol information (except for performance data; see the `--performance` option). However, when the `--protocol` option is specified, the automatic gathering of all protocol information is turned off, and only the specified type of protocol information is gathered. The following values for *ProtocolType* are accepted:

```
smb
nfs
object
authentication
ces
core
none
```

--timeout *Seconds*
Specifies the timeout value, in seconds, for all commands.

--purge-files *KeepNumberOfDaysBack*
Specifies that large debug files is deleted from the cluster nodes based on the *KeepNumberOfDaysBack* value. If `0` is specified, all of the large debug files is deleted. If a value greater than `0` is specified, large debug files that are older than the number of days specified are deleted. For example, if the value 2 is specified, the previous two days of large debug files are retained.

This option is not compatible with many of the `gpfs.snap` options because it only removes files and does not collect any `gpfs.snap` data.

--hadoop
Specifies that Hadoop data is to be gathered.

Use the `-z` option to generate a non-master snapshot. This option is useful if there are many nodes on which to take a snapshot, and only one master snapshot is needed. For a GPFS problem within a large cluster (hundreds or thousands of nodes), one strategy might call for a single master snapshot (one invocation of `gpfs.snap` with no options), and multiple non-master snapshots (multiple invocations of `gpfs.snap` with the `-z` option).

Use the `-N` option to obtain `gpfs.snap` data from multiple nodes in the cluster. When the `-N` option is used, the `gpfs.snap` command takes non-master snapshots of all the nodes that are specified with this option and a master snapshot of the node on which it was issued.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `gpfs.snap` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see the topic *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

Examples

1. To collect **gpfs.snap** on all nodes with the default data, issue the following command:

```
c34f2n03:# gpfs.snap
gpfs.snap started at Fri Mar 22 13:16:12 EDT 2019.
Gathering common data...
Gathering Linux specific data...
Gathering extended network data...
Gathering local callhome data...
Gathering local sysmon data...
Gathering trace reports and internal dumps...
Gathering Transparent Cloud Tiering data at level BASIC...
gpfs.snap:  The Transparent Cloud Tiering snap file was not located on the node c34f2n03.gpfs.net
Gathering cluster wide sysmon data...
Gathering cluster wide callhome data...
gpfs.snap:  Spawning remote gpfs.snap calls. Master is c34f2n03.
This may take a while.

Copying file /tmp/gpfs.snapOut/27348/gpfs.snap.c13c1apv7_0322131503.out.tar.gz from c13c1apv7.gpfs.net ...
gpfs.snap.c13c1apv7_0322131503.out.tar.gz        100% 7732KB   7.6MB/s   00:00
Successfully copied file /tmp/gpfs.snapOut/27348/gpfs.snap.c13c1apv7_0322131503.out.tar.gz from
     c13c1apv7.gpfs.net.
Gathering cluster wide protocol data...
Gathering waiters from all nodes...
Gathering mmfs.logs from all nodes.  This may take a while...
All data has been collected.  Processing collected data.
  This may take a while...
Packaging master node data...
Writing * to file /tmp/gpfs.snapOut/27348/collect/gpfs.snap.c34f2n03_master_0322131612.out.tar.gz
Packaging all data.
Writing . to file /tmp/gpfs.snapOut/27348/all.0322131612.tar
gpfs.snap completed at Fri Mar 22 13:17:55 EDT 2019
################################################################################
Send file /tmp/gpfs.snapOut/27348/all.0322131612.tar to IBM Service
Examine previous messages to determine additional required data.
################################################################################
```

   After this command has completed, send the tar file (highlighted) to IBM service.

2. The following example collects **gpfs.snap** data on specific nodes and provides an output directory:

```
./gpfs.snap -N c34f2n03.gpfs.net,c13c1apv7.gpfs.net -d /tmp/snap_outdir
gpfs.snap started at Fri Mar 22 15:54:35 EDT 2019.
Gathering common data...
Gathering Linux specific data...
Gathering extended network data...
Gathering local callhome data...
Gathering local sysmon data...
Gathering trace reports and internal dumps...
Gathering Transparent Cloud Tiering data at level BASIC...
gpfs.snap:  The Transparent Cloud Tiering snap file was not located on the node
     c34f2n03.gpfs.net
Gathering cluster wide sysmon data...
Gathering cluster wide callhome data...
gpfs.snap:  Spawning remote gpfs.snap calls. Master is c34f2n03.
This may take a while.

Copying file /tmp/snap_outdir/gpfs.snap.c13c1apv7_0322155324.out.tar.gz from c13c1apv7.gpfs.net
     ...
gpfs.snap.c13c1apv7_0322155324.out.tar.gz     100% 7720KB   7.5MB/s   00:00
Successfully copied file /tmp/snap_outdir/gpfs.snap.c13c1apv7_0322155324.out.tar.gz from
     c13c1apv7.gpfs.net.
Gathering cluster wide protocol data...
Gathering waiters from all nodes...
Gathering mmfs.logs from all nodes.  This may take a while...
All data has been collected.  Processing collected data.
  This may take a while...
Packaging master node data...
Writing * to file /tmp/snap_outdir/collect/gpfs.snap.c34f2n03_master_0322155435.out.tar.gz

Packaging all data.
Writing . to file /tmp/snap_outdir/all.0322155435.tar
gpfs.snap completed at Fri Mar 22 15:56:11 EDT 2019
################################################################################
Send file /tmp/snap_outdir/all.0322155435.tar to IBM Service
Examine previous messages to determine additional required data.
################################################################################
```

Location

/usr/lpp/mmfs/bin

mmaddcallback command

Registers a user-defined command that GPFS will execute when certain events occur.

Synopsis

```
mmaddcallback CallbackIdentifier --command CommandPathname
              --event Event[,Event...] [--priority Value]
              [--async | --sync [--timeout Seconds] [--onerror Action]]
              [-N {Node[,Node...] | NodeFile | NodeClass}]
              [--parms ParameterString ...]
```

or

```
mmaddcallback {-S Filename | --spec-file Filename}
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmaddcallback` command to register a user-defined command that GPFS executes when certain events occur.

The callback mechanism is intended to provide notifications when node and cluster events occur. Invoking complex or long-running commands, or commands that involve GPFS files, might cause unexpected and undesired results, including loss of file system availability. This is particularly true when the `--sync` option is specified.

Note: For documentation about local events (callbacks) and variables for IBM Spectrum Scale RAID, see the separate publication *IBM Spectrum Scale RAID: Administration*.

Parameters

CallbackIdentifier
Specifies a user-defined unique name that identifies the callback. It can be up to 255 characters long. It cannot contain special characters (for example, a colon, semicolon, blank, tab, or comma) and it cannot start with the letters gpfs or mm (which are reserved for GPFS internally defined callbacks).

--command *CommandPathname*
Specifies the full path name of the executable to run when the event occurs. On Windows, *CommandPathname* must be a Korn shell script because it will be invoked in the Cygwin `ksh` environment.

The executable called by the callback facility must be installed on all nodes on which the callback can be triggered. Place the executable in a local file system (not in a GPFS file system) so that it is accessible even when the GPFS file system is unavailable.

--event *Event*[,*Event*...]
Specifies a list of events that trigger the callback. The value defines when the callback is invoked. There are two kinds of events: global events and local events. A global event triggers a callback on all nodes in the cluster, such as a **nodeLeave** event, which informs all nodes in the cluster that a node has failed. A local event triggers a callback only on the node on which the event occurred, such as mounting a file system on one of the nodes.

Table 8 on page 17 lists the supported global events and their parameters.

Table 9 on page 18 lists the supported local events and their parameters.

Local events for IBM Spectrum Scale RAID are documented in *IBM Spectrum Scale RAID: Administration*.

--priority ***Value***
Specifies a floating point number that controls the order in which callbacks for a given event are run. Callbacks with a smaller numerical value are run before callbacks with a larger numerical value. Callbacks that do not have an assigned priority are run last. If two callbacks have the same priority, the order in which they are run is undetermined.

--async | --sync [--timeout ***Seconds***] **[--onerror** ***Action***]
Specifies whether GPFS will wait for the user program to complete and for how long it will wait. The default is `--async` (GPFS invokes the command asynchronously). `--onerror` *Action* specifies one of the following actions that GPFS is to take if the callback command returns a nonzero error code:

continue
GPFS ignores the result from executing the user-provided command. This is the default.

quorumLoss
The node executing the user-provided command will voluntarily resign as, or refrain from taking over as, cluster manager. This action is valid only in conjunction with the `tiebreakerCheck` event.

shutdown
GPFS will be shut down on the node executing the user-provided command.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Defines the set of nodes on which the callback is invoked. For global events, the callback is invoked only on the specified set of nodes. For local events, the callback is invoked only if the node on which the event occurred is one of the nodes specified by the `-N` option. The default is `-N all`. For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in *IBM Spectrum Scale: Administration Guide*.

This command does not support a *NodeClass* of `mount`.

--parms ***ParameterString*** **...**
Specifies parameters to be passed to the executable specified with the `--command` parameter. The `--parms` parameter can be specified multiple times.

When the callback is invoked, the combined parameter string is tokenized on white-space boundaries. Constructs of the form *%name* and *%name.qualifier* are assumed to be GPFS variables and are replaced with their appropriate values at the time of the event. If a variable does not have a value in the context of a particular event, the string `UNDEFINED` is returned instead.

GPFS recognizes the following variables:

%blockLimit
Specifies the current hard quota limit in KB.

%blockQuota
Specifies the current soft quota limit in KB.

%blockUsage
Specifies the current usage in KB for quota-related events.

%ccrObjectName
Specifies the name of the modified object.

%ccrObjectValue
Specifies the value of the modified object.

%ccrObjectVersion
Specifies the version of the modified object.

%clusterManager[.*qualifier*]
Specifies the current cluster manager node.

%clusterName
Specifies the name of the cluster where this callback was triggered.

%ckDataLen
Specifies the length of data involved in a checksum mismatch.

%ckErrorCountClient
Specifies the cumulative number of errors for the client side in a checksum mismatch.

%ckErrorCountNSD
Specifies the cumulative number of errors for the NSD side in a checksum mismatch.

%ckErrorCountServer
Specifies the cumulative number of errors for the server side in a checksum mismatch.

%ckNSD
Specifies the NSD involved.

%ckOtherNode
Specifies the IP address of the other node in an NSD checksum event.

%ckReason
Specifies the reason string indicating why a checksum mismatch callback was invoked.

%ckReportingInterval
Specifies the error-reporting interval in effect at the time of a checksum mismatch.

%ckRole
Specifies the role (client or server) of a GPFS node.

%ckStartSector
Specifies the starting sector of a checksum mismatch.

%daName
Specifies the name of the declustered array involved.

%daRemainingRedundancy
Specifies the remaining fault tolerance in a declustered array.

%diskName
Specifies a disk or a comma-separated list of disk names for which this callback is triggered.

%downNodes[.*qualifier*]
Specifies a comma-separated list of nodes that are currently down. Only nodes local to the given cluster are listed. Nodes which are in a remote cluster but have temporarily joined the cluster are not included.

%eventName
Specifies the name of the event that triggered this callback.

%eventNode[.*qualifier*]
Specifies a node or comma-separated list of nodes on which this callback is triggered. Note that the list might include nodes which are not local to the given cluster, but have temporarily joined the cluster to mount a file system provided by the local cluster. Those remote nodes could leave the cluster if there is a node failure or if the file systems are unmounted.

%eventTime
Specifies the time of the event that triggered this callback.

%filesLimit
Specifies the current hard quota limit for the number of files.

%filesQuota
Specifies the current soft quota limit for the number of files.

%filesUsage
Specifies the current number of files for quota-related events.

%filesetName
Specifies the name of a fileset for which the callback is being executed.

%filesetSize
Specifies the size of the fileset.

%fsErr
Specifies the file system structure error code.

%fsName
Specifies the file system name for file system events.

%hardLimit
Specifies the hard limit for the block.

%homeServer
Specifies the name of the home server.

%inodeLimit
Specifies the hard limit of the inode.

%inodeQuota
Specifies the soft limit of the inode.

%inodeUsage
Specifies the total number of files in the fileset.

%myNode[.*qualifier*]
Specifies the node where callback script is invoked.

%nodeName
Specifies the node name to which the request is sent.

%nodeNames
Specifies a space-separated list of node names to which the request is sent.

%pcacheEvent
Specifies the pcache related events.

%pdFru
Specifies the FRU (field replaceable unit) number of the pdisk.

%pdLocation
The physical location code of a pdisk.

%pdName
The name of the pdisk involved.

%pdPath
The block device path of the pdisk.

%pdPriority
The replacement priority of the pdisk.

%pdState
The state of the pdisk involved.

%pdWwn
The worldwide name of the pdisk.

%prepopAlreadyCachedFiles
Specifies the number of files that are cached. These number of files are not read into cache because data is same between cache and home.

%prepopCompletedReads
Specifies the number of reads executed during a prefetch operation.

%prepopData
Specifies the total data read from the home as part of a prefetch operation.

%prepopFailedReads
Specifies the number of files for which prefetch failed. Messages are logged to indicate the failure. However, there is no indication about the file names that failed to read.

%quorumNodes[.*qualifier*]
Specifies a comma-separated list of quorum nodes.

%quotaEventType
Specifies either the **blockQuotaExceeded** event or the **inodeQuotaExceeded** event. These events are related to soft quota limit being exceeded,

%quotaID
Specifies the numerical ID of the quota owner (UID, GID, or fileset ID).

%quotaOwnerName
Specifies the name of the quota owner (user name, group name, or fileset name).

%quotaType
Specifies the type of quota for quota-related events. Possible values are USR, GRP, or FILESET.

%reason
Specifies the reason for triggering the event. For the preUnmount and unmount events, the possible values are normal and forced. For the preShutdown and shutdown events, the possible values are normal and abnormal. For all other events, the value is UNDEFINED.

%requestType
Specifies the type of request to send to the target nodes.

%rgCount
The number of recovery groups involved.

%rgErr
A code from a recovery group, where 0 indicates no error.

%rgName
The name of the recovery group involved.

%rgReason
The reason string indicating why a recovery group callback was invoked.

%senseDataFormatted
Sense data for the specific fileset structure error in a formatted string output.

%senseDataHex
Sense data for the specific fileset structure error in Big endian hex output.

%snapshotID
Specifies the identifier of the new snapshot.

%snapshotName
Specifies the name of the new snapshot.

%softLimit
Specifies the soft limit of the block.

%storagePool
Specifies the storage pool name for space-related events.

%upNodes[.*qualifier*]
Specifies a comma-separated list of nodes that are currently up. Only nodes local to the given cluster are listed. Nodes which are in a remote cluster but have temporarily joined the cluster are not included.

%userName
Specifies the user name.

%waiterLength
Specifies the length of the waiter in seconds.

Variables recognized by IBM Spectrum Scale RAID are documented in *IBM Spectrum Scale RAID: Administration*.

Variables that represent node identifiers accept an optional qualifier that can be used to specify how the nodes are to be identified. When specifying one of these optional qualifiers, separate it from the variable with a period, as shown here:

```
variable.qualifier
```

The value for *qualifier* can be one of the following:

ip
Specifies that GPFS should use the nodes' IP addresses.

name
Specifies that GPFS should use fully-qualified node names. This is the default.

shortName
Specifies that GPFS should strip the domain part of the node names.

Events and supported parameters

Table 8. Global events and supported parameters

Global event	Supported parameters
afmFilesetExpired Triggered when the contents of a fileset expire either as a result of the fileset being disconnected for the expiration timeout value or when the fileset is marked as expired using the AFM administration commands.	**%fsName %filesetName %pcacheEvent %homeServer %reason**
afmFilesetUnexpired Triggered when the contents of a fileset become unexpired either as a result of the reconnection to home or when the fileset is marked as unexpired using the AFM administration commands.	**%fsName %filesetName %pcacheEvent %homeServer %reason**
clusterManagerTakeOver Triggered when a new cluster manager node is elected. This happens when a cluster first starts up or when the current cluster manager fails or resigns and a new node takes over as cluster manager.	N/A
nodeJoin Triggered when one or more nodes join the cluster.	**%eventNode**
nodeLeave Triggered when one or more nodes leave the cluster.	**%eventNode**
quorumReached Triggered when a quorum has been established in the GPFS cluster. This event is triggered only on the cluster manager, not on all the nodes in the cluster.	**%quorumNodes**
quorumLoss Triggered when quorum has been lost in the GPFS cluster.	N/A
quorumNodeJoin Triggered when one or more quorum nodes join the cluster.	**%eventNode**
quorumNodeLeave Triggered when one or more quorum nodes leave the cluster.	**%eventNode**

Table 8. Global events and supported parameters (continued)	
Global event	**Supported parameters**
quotaExceeded Triggered when quota clients fail to allocate disk space because the quota hard limit (for either files or blocks) is reached. This event is triggered on all nodes.	**%fsName %filesetName %quotaId %quotaType %quotaOwnerName %eventTime %quotaEventType**

Table 9. Local events and supported parameters	
Local event	**Supported parameters**
afmCmdRequeued Triggered during replication when messages are queued up again because of errors. These messages are retried after 15 minutes.	**%fsName %filesetName %pcacheEvent %homeServer %reason**
afmFilesetUnmounted Triggered when the fileset is moved to an Unmounted state because NFS server is not reachable or remote cluster mount is not available for GPFS Native protocol.	**%fsName %filesetName %pcacheEvent %homeServer %reason**
afmHomeConnected Triggered when a gateway node connects to the afmTarget of the fileset that it is serving. This event is local on gateway nodes.	**%fsName %filesetName %pcacheEvent %homeServer %reason**
afmHomeDisconnected Triggered when a gateway node gets disconnected from the afmTarget of the fileset that it is serving. This event is local on gateway nodes.	**%fsName %filesetName %pcacheEvent %homeServer %reason**
afmManualResyncComplete Triggered when a manual resync is completed.	**%fsName %filesetName %reason**
afmPrepopEnd Triggered when all the files specified by a prefetch operation have been completed. This event is local on a gateway node.	**%fsName %filesetName %prepopCompletedReads %prepopFailedReads %prepopAlreadyCachedFiles %prepopData**
afmQueueDropped Triggered when replication encounters an issue that cannot be corrected. After the queue is dropped, next recovery action attempts to fix the error and continue to replicate.	**%fsName %filesetName %pcacheEvent %homeServer %reason**
afmRecoveryFail Triggered when recovery fails. The recovery action is retried after 300 seconds. If recovery keeps failing, fileset is moved to a resync state if the fileset mode allows it.	**%fsName %filesetName %pcacheEvent %homeServer %reason**

Table 9. Local events and supported parameters (continued)

Local event	Supported parameters
afmRecoveryStart Triggered when AFM recovery starts. This event is local on gateway nodes.	**%fsName %filesetName %pcacheEvent %homeServer %reason**
afmRecoveryEnd Triggered when AFM recovery ends. This event is local on gateway nodes.	**%fsName %filesetName %pcacheEvent %homeServer %reason**
afmRPOMiss Triggered when Recovery Point Objective (RPO) is missed on DR primary filesets, RPO Manager keeps retrying the snapshots. This event occurs when there is lot of data to replicate for the RPO snapshot to be taken or there is an error such as, deadlock and recovery keeps failing.	**%fsName %filesetName %pcacheEvent %homeServer %reason**
afmFilesetCreate Triggered when an AFM fileset is created.	**%fsName %filesetName %pcacheEvent %homeServer %reason**
afmFilesetLink Triggered when an AFM fileset is linked.	**%fsName %filesetName %pcacheEvent %homeServer %reason**
afmFilesetChange Triggered when an AFM fileset is changed. If a fileset is renamed the new name is part of %reason.	**%fsName %filesetName %pcacheEvent %homeServer %reason**
afmFilesetUnlink Triggered when a AFM fileset is linked.	**%fsName %filesetName %pcacheEvent %homeServer %reason**
afmFilesetDelete Triggered when an AFM fileset is deleted.	**%fsName %filesetName %pcacheEvent %homeServer %reason**
ccrFileChange Triggered when CCR `fput` operation takes place.	**%ccrObjectName %ccrObjectVersion**
ccrVarChange Triggered when CCR `vput` operation takes place.	**%ccrObjectName %ccrObjectValue %ccrObjectVersion**
daRebuildFailed The `daRebuildFailed` callback is generated when the spare space in a declustered array has been exhausted, and vdisk tracks involving damaged pdisks can no longer be rebuilt. The occurrence of this event indicates that fault tolerance in the declustered array has become degraded and that disk maintenance should be performed immediately. The `daRemainingRedundancy` parameter indicates how much fault tolerance remains in the declustered array.	**%myNode %rgName %daName %daRemainingRedundancy**

Table 9. Local events and supported parameters (continued)

Local event	Supported parameters
deadlockDetected Triggered when a node detects a potential deadlock. If the exit code of the registered callback for this event is 1, debug data will not be collected. See the `/usr/lpp/mmfs/samples/deadlockdetected.sample` file for an example of using the `deadlockDetected` event.	**%eventName %myNode %waiterLength**
deadlockOverload Triggered when an overload event occurs. The event is local to the node detecting the overload condition.	**%eventName %nodeName**
diskFailure Triggered on the file system manager when the status of a disk in a file system changes to down.	**%eventName %diskName %fsName**
diskIOErr Triggered when an NSD server node detects a problem in a locally attached disk. For more information, see the topic *Periodic disk I/O check* in the *IBM Spectrum Scale: Administration Guide*.	**%nodeName %diskName**

Table 9. Local events and supported parameters (continued)

Local event	Supported parameters
diskIOHang Triggered when the GPFS daemon detects that a local I/O request has been pending in the kernel for more than five minutes. The event occurs only once, even if there are multiple concurrent long I/O waiters. This event applies only to disks that a node is directly attached to. With this callback event you can add notification and data collection scripts to isolate the reason for a long I/O wait. This callback is not supported in the Microsoft Windows environment. The event is generated regardless of the value of the `panicOnIOHang` attribute of the `mmchconfig command`. If an I/O hang occurs when the `diskIOHang` event is configured and the `panicOnIOHang` attribute of the `mmchconfig` command is set to `yes`, the result depends upon whether the `--async` or `--sync` parameter was specified for the `diskIOHang` event: • With `--async`, the GPFS daemon waits five seconds for the `diskIOHang` scripts to run and then panics the node kernel. • With `--sync`, the GPFS daemon waits for the `diskIOHang` scripts to run to completion before it panics the node kernel.	**%fsName %diskName**
filesetLimitExceeded Triggered when the file system manager detects that a fileset quota has been exceeded. This is a variation of `softQuotaExceeded` that applies only to fileset quotas. It exists only for compatibility (and can be deleted in a future version); therefore, using `softQuotaExceeded` is recommended instead.	**%filesetName %fsName %filesetSize %softLimit %hardLimit %inodeUsage %inodeQuota %inodeLimit %quotaEventType**
fsstruct Triggered when the file system manager detects a file system structure (FS Struct) error. For more information about FS Struct errors, see “mmfsck command” on page 410 and the following topics in *IBM Spectrum Scale: Problem Determination Guide*: • *MMFS_FSSTRUCT* • *Reliability, Availability, and Serviceability (RAS) events* • *Information to collect before contacting the IBM Support Center*	**%fsName %fsErr %senseDataFormatted %senseDataHex**

Table 9. Local events and supported parameters (continued)

Local event	Supported parameters
lowDiskSpace Triggered when the file system manager detects that disk space usage has reached the high occupancy threshold that is specified in the current policy rule. The event is generated every two minutes until the condition no longer exists. For more information, see the topic *Using thresholds with external pools* in the *IBM Spectrum Scale: Administration Guide*.	**%storagePool %fsName**
noDiskSpace Triggered when the file system encounters a disk, or storage pool that has run out of space or an inodespace has run out of inodes. An inode space can be an entire file system or an independent fileset. Use the `noSpaceEventInterval` configuration attribute of the `mmchconfig` command to control the time interval between two `noDiskSpace` events. The default value is 120 seconds. When a storage pool runs out of disk space, `%reason` is "diskspace", `%storagePool` is the name of the pool that ran out of disk space, and `%filesetName` is "UNDEFINED". When a fileset runs out of inode space, `%reason` is "inodespace", `%filesetName` is the name of the independent fileset that owns the affected inode space, and `%storagePool` is "UNDEFINED".	**%storagePool %fsName %reason %filesetName**
nsdCksumMismatch The `nsdCksumMismatch` callback is generated whenever transmission of vdisk data by the NSD network layer fails to verify the data checksum. This can indicate problems in the network between the GPFS client node and a recovery group server. The first error between a given client and server generates the callback; subsequent callbacks are generated for each `ckReportingInterval` occurrence.	**%myNode %ckRole %ckOtherNode %ckNSD %ckReason %ckStartSector %ckDataLen %ckErrorCountClient %ckErrorCountServer %ckErrorCountNSD %ckReportingInterval**
pdFailed The `pdFailed` callback is generated whenever a pdisk in a recovery group is marked as `dead`, `missing`, `failed`, or `readonly`.	**%myNode %rgName %daName %pdName %pdLocation %pdFru %pdWwn %pdState**

Table 9. Local events and supported parameters (continued)

Local event	Supported parameters
pdPathDown The pdPathDown callback is generated whenever one of the block device paths to a pdisk disappears or becomes inoperative. The occurrence of this event can indicate connectivity problems with the JBOD array in which the pdisk resides.	**%myNode %rgName %daName %pdName %pdLocation %pdFru %pdWwn %pdPath**
pdReplacePdisk The pdReplacePdisk callback is generated whenever a pdisk is marked for replacement according to the replace threshold setting of the declustered array in which it resides.	**%myNode %rgName %daName %pdName %pdLocation %pdFru %pdWwn %pdState %pdPriority**
pdRecovered The pdRecovered callback is generated whenever a missing pdisk is rediscovered. The following parameters are available to this callback: %myNode, %rgName, %daName, %pdName, %pdLocation, %pdFru, and %pdWwn.	**%myNode %rgName %daName %pdName %pdLocation %pdFru %pdWwn**
preMount, preUnmount, mount, unmount These events are triggered when a file system is about to be mounted or unmounted or has been mounted or unmounted successfully. These events are generated for explicit **mount** and **unmount** commands, a remount after GPFS recovery and a forced **unmount** when GPFS panics and shuts down.	**%fsName %reason**
preRGRelinquish The preRGRelinquish callback is invoked on a recovery group server prior to relinquishing service of recovery groups. The rgName parameter can be passed into the callback as the keyword value _ALL_, indicating that the recovery group server is about to relinquish service for all recovery groups it is serving; the rgCount parameter will be equal to the number of recovery groups being relinquished. Additionally, the callback will be invoked with the rgName of each individual recovery group and an rgCount of 1 whenever the server relinquishes serving recovery group rgName.	**%myNode %rgName %rgErr %rgCount %rgReason**

Table 9. Local events and supported parameters (continued)

Local event	Supported parameters
preRGTakeover The `preRGTakeover` callback is invoked on a recovery group server prior to attempting to open and serve recovery groups. The `rgName` parameter can be passed into the callback as the keyword value `_ALL_`, indicating that the recovery group server is about to open multiple recovery groups; this is typically at server startup, and the parameter `rgCount` will be equal to the number of recovery groups being processed. Additionally, the callback will be invoked with the `rgName` of each individual recovery group and an `rgCount` of 1 whenever the server checks to determine whether it should open and serve recovery group `rgName`.	**%myNode %rgName %rgErr %rgCount %rgReason**
preShutdown Triggered when GPFS detects a failure and is about to shut down.	**%reason**
preStartup Triggered after the GPFS daemon completes its internal initialization and joins the cluster, but before the node runs recovery for any file systems that were already mounted, and before the node starts accepting user initiated sessions.	N/A
postRGRelinquish The `postRGRelinquish` callback is invoked on a recovery group server after it has relinquished serving recovery groups. If multiple recovery groups have been relinquished, the callback will be invoked with `rgName` keyword `_ALL_` and an `rgCount` equal to the total number of involved recovery groups. The callback will also be triggered for each individual recovery group.	**%myNode %rgName %rgErr %rgCount %rgReason**
postRGTakeover The `postRGTakeover` callback is invoked on a recovery group server after it has checked, attempted, or begun to serve a recovery group. If multiple recovery groups have been taken over, the callback will be invoked with `rgName` keyword `_ALL_` and an `rgCount` equal to the total number of involved recovery groups. The callback will also be triggered for each individual recovery group.	**%myNode %rgName %rgErr %rgCount %rgReason**

Table 9. Local events and supported parameters (continued)

Local event	Supported parameters
rgOpenFailed The `rgOpenFailed` callback will be invoked on a recovery group server when it fails to open a recovery group that it is attempting to serve. This might be due to loss of connectivity to some or all of the disks in the recovery group; the `rgReason` string will indicate why the recovery group could not be opened.	**%myNode %rgName %rgErr %rgReason**
rgPanic The `rgPanic` callback will be invoked on a recovery group server when it is no longer able to continue serving a recovery group. This might be due to loss of connectivity to some or all of the disks in the recovery group; the `rgReason` string will indicate why the recovery group can no longer be served.	**%myNode %rgName %rgErr %rgReason**
sendRequestToNodes Triggered when a node sends a request for collecting expel-related debug data. For this event, the `%requestType` is `requestExpelData`.	**%eventName %requestType %nodeNames**
shutdown Triggered when GPFS completes the shutdown.	**%reason**
snapshotCreated Triggered after a snapshot is created, and run before the file system is resumed. This event helps correlate the timing of DMAPI events with the creation of a snapshot. GPFS must wait for `snapshotCreated` to exit before it resumes the file system, so the ordering of DMAPI events and snapshot creation is known. The `%filesetName` is the name of the fileset whose snapshot was created. For file system level snapshots that affect all filesets, %filesetName is set to `global`.	**%snapshotID %snapshotName %fsName %filesetName**
softQuotaExceeded Triggered when the file system manager detects that a soft quota limit (for either files or blocks) has been exceeded. This event is triggered only on the file system manager. Therefore, this event must be handled on all manager nodes.	**%fsName %filesetName %quotaId %quotaType %quotaOwnerName %blockUsage %blockQuota %blockLimit %filesUsage %filesQuota %filesLimit**

Table 9. Local events and supported parameters (continued)

Local event	Supported parameters
startup Triggered after a successful GPFS startup before the node is ready for user initiated sessions. After this event is triggered GPFS proceeds to finish starting including mounting all file systems defined to mount on startup.	N/A
tiebreakerCheck Triggered when the cluster manager detects a lease timeout on a quorum node before GPFS runs the algorithm that decides if the node will remain in the cluster. This event is generated only in configurations that use tiebreaker disks. **Note:** Before you add or delete the **tiebreakerCheck** event, you must stop the GPFS daemon on all the nodes in the cluster.	N/A
traceConfigChanged Triggered when GPFS tracing configuration is changed.	N/A
usageUnderSoftQuota Triggered when the file system manager detects that quota usage has dropped below soft limits and grace time is reset.	**%fsName %filesetName %fsName %quotaId %quotaType %quotaOwnerName %blockUsage %blockQuota %blockLimit %filesUsage %filesQuota %filesLimit**

Options

-S *Filename* | --spec-file *Filename*
Specifies a file with multiple callback definitions, one per line. The first token on each line must be the callback identifier.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmaddcallback` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To register command **/tmp/myScript** to run after GPFS startup, issue the following command:

```
# mmaddcallback test1 --command=/tmp/myScript --event startup
```

A sample output is as follows:

```
mmaddcallback: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

2. To register a callback on the NFS servers to export or to unexport a particular file system after it has been mounted or before it has been unmounted, issue the following command:

```
# mmaddcallback NFSexport --command /usr/local/bin/NFSexport --event mount,preUnmount -N
nfsserver1,
nfsserver2 --parms "%eventName %fsName" --parms "%eventName %fsName"
```

A sample output is as follows:

```
mmaddcallback: 6027-1371 Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

See also

- “mmdelcallback command” on page 362
- “mmlscallback command” on page 490

Location

/usr/lpp/mmfs/bin

mmadddisk command

Adds disks to a GPFS file system.

Synopsis

```
mmadddisk Device {"DiskDesc[;DiskDesc...]" | -F StanzaFile} [-a] [-r [--strict]]
  [-v {yes | no}] [-N {Node[,Node...] | NodeFile | NodeClass}]
  [--qos QOSClass]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmadddisk` command to add disks to a GPFS file system. When the `-r` flag is specified, the command rebalances an existing file system after it adds the disks. The command does not require the file system to be mounted. The file system can be in use.

The actual number of disks in your file system might be constrained by products other than GPFS that you installed. See to the individual product documentation.

To add disks to a GPFS file system, first decide which of the following two tasks you want to perform:

1. Create new disks with the `mmcrnsd` command.

   In this case, you must also decide whether to create a new set of NSD and pools stanzas or use the rewritten NSD and pool stanzas that the **mmcrnsd** command produces. In a rewritten file, the disk usage, failure group, and storage pool values are the same as the values that are specified in the **mmcrnsd** command.
2. Select disks no longer in use in any file system. To display the disks that are not in use, run the following command:

   ```
   mmlsnsd -F
   ```

Earlier versions of the product allowed specifying disk information with colon-separated disk descriptors. Those disk descriptors are no longer supported.

Note: If `mmadddisk` fails with a NO_SPACE error, try one of the following actions:

- Rebalance the file system.
- Run the command `mmfsck -y` to deallocate unreferenced subblocks.
- Create a pool with larger disks and move data from the old pool to the new one.

Parameters

Device
The device name of the file system to which the disks are added. File system names need not be fully qualified. `fs0` is as acceptable as `/dev/fs0`.

This parameter must be first.

DiskDesc
A descriptor for each disk to be added. Each descriptor is delimited by a semicolon (`;`) and the entire list must be enclosed in quotation marks (`'` or `"`). The use of disk descriptors is discouraged.

-F *StanzaFile*
Specifies a file that contains the NSD stanzas and pool stanzas for the disks to be added to the file system.

NSD stanzas have the following format:

```
%nsd:
  nsd=NsdName
  usage={dataOnly | metadataOnly | dataAndMetadata | descOnly}
  failureGroup=FailureGroup
  pool=StoragePool
  servers=ServerList
  device=DiskName
  thinDiskType={no | nvme | scsi | auto}
```

where:

nsd=*NsdName*
The name of an NSD previously created by the `mmcrnsd` command. For a list of available disks, run the `mmlsnsd -F` command. This clause is mandatory for the `mmadddisk` command.

usage={dataOnly | metadataOnly | dataAndMetadata | descOnly}
Specifies the type of data to be stored on the disk:

dataAndMetadata
Indicates that the disk contains both data and metadata. This value is the default for disks in the system pool.

dataOnly
Indicates that the disk contains data and does not contain metadata. This value is the default for disks in storage pools other than the system pool.

metadataOnly
Indicates that the disk contains metadata and does not contain data.

descOnly
Indicates that the disk contains no data and no file metadata. IBM Spectrum Scale uses this type of disk primarily to keep a copy of the file system descriptor. It can also be used as a third failure group in certain disaster recovery configurations. For more information, see the topic *Synchronous mirroring utilizing GPFS replication* in the *IBM Spectrum Scale: Administration Guide*.

failureGroup=*FailureGroup*
Identifies the failure group to which the disk belongs. A failure group identifier can be a simple integer or a topology vector that consists of up to three comma-separated integers. The default is -1, which indicates that the disk has no point of failure in common with any other disk.

GPFS uses this information during data and metadata placement to ensure that no two replicas of the same block can become unavailable due to a single failure. All disks that are attached to the same NSD server or adapter must be placed in the same failure group.

If the file system is configured with data replication, all storage pools must have two failure groups to maintain proper protection of the data. Similarly, if metadata replication is in effect, the system storage pool must have two failure groups.

Disks that belong to storage pools in which write affinity is enabled can use topology vectors to identify failure domains in a shared-nothing cluster. Disks that belong to traditional storage pools must use simple integers to specify the failure group.

pool=*StoragePool*
Specifies the storage pool to which the disk is to be assigned. If this name is not provided, the default is `system`.

Only the system storage pool can contain `metadataOnly`, `dataAndMetadata`, or `descOnly` disks. Disks in other storage pools must be `dataOnly`.

servers=*ServerList*
A comma-separated list of NSD server nodes. This clause is ignored by the `mmadddisk` command.

device=*DiskName*
The block device name of the underlying disk device. This clause is ignored by the `mmadddisk` command.

thinDiskType={no | nvme | scsi | auto}
Specifies the space reclaim disk type:

no
The disk does not support space reclaim. This value is the default.

nvme
The disk is a TRIM capable NVMe device that supports the `mmreclaimspace` command.

scsi
The disk is a thin provisioned SCSI disk that supports the `mmreclaimspace` command.

auto
The type of the disk is either `nvme` or `scsi`. IBM Spectrum Scale will try to detect the actual disk type automatically. To avoid problems, you should replace `auto` with the correct disk type, `nvme` or `scsi`, as soon as you can.

Note: In 5.0.5, the space reclaim auto-detection is enhanced. It is encouraged to use the auto key-word after your cluster is upgraded to 5.0.5.

For more information, see the topic *IBM Spectrum Scale with data reduction storage devices* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Note: An NSD belonging to a tiebreaker disk is not allowed to be added to a file system if NSD format conversion is required.

Pool stanzas have this format:

```
%pool:
  pool=StoragePoolName
  blockSize=BlockSize
  usage={dataOnly | metadataOnly | dataAndMetadata}
  layoutMap={scatter | cluster}
  allowWriteAffinity={yes | no}
  writeAffinityDepth={0 | 1 | 2}
  blockGroupFactor=BlockGroupFactor
```

where:

pool=*StoragePoolName*
Is the name of a storage pool.

blockSize=*BlockSize*
Specifies the block size of the disks in the storage pool.

usage={dataOnly | metadataOnly | dataAndMetadata}
Specifies the type of data to be stored in the storage pool:

dataAndMetadata
Indicates that the disks in the storage pool contain both data and metadata. This is the default for disks in the system pool.

dataOnly
Indicates that the disks contain data and do not contain metadata. This is the default for disks in storage pools other than the system pool.

metadataOnly
Indicates that the disks contain metadata and do not contain data.

layoutMap={scatter | cluster}
Specifies the block allocation map type. When allocating blocks for a given file, GPFS first uses a round-robin algorithm to spread the data across all disks in the storage pool. After a disk is selected, the location of the data block on the disk is determined by the block allocation map type. If `cluster` is specified, GPFS attempts to allocate blocks in clusters. Blocks that belong to a particular file are kept adjacent to each other within each cluster. If `scatter` is specified, the location of the block is chosen randomly.

The `cluster` allocation method may provide better disk performance for some disk subsystems in relatively small installations. The benefits of clustered block allocation diminish when the

number of nodes in the cluster or the number of disks in a file system increases, or when the file system's free space becomes fragmented. The `cluster` allocation method is the default for GPFS clusters with eight or fewer nodes and for file systems with eight or fewer disks.

The `scatter` allocation method provides more consistent file system performance by averaging out performance variations due to block location (for many disk subsystems, the location of the data relative to the disk edge has a substantial effect on performance). This allocation method is appropriate in most cases and is the default for GPFS clusters with more than eight nodes or file systems with more than eight disks.

The block allocation map type cannot be changed after the storage pool has been created.

allowWriteAffinity={yes | no}
Indicates whether the File Placement Optimizer (FPO) feature is to be enabled for the storage pool. For more information on FPO, see the *File Placement Optimizer* section in the *IBM Spectrum Scale: Administration Guide*.

writeAffinityDepth={0 | 1 | 2}
Specifies the allocation policy to be used by the node writing the data.

A write affinity depth of 0 indicates that each replica is to be striped across the disks in a cyclical fashion with the restriction that no two disks are in the same failure group. By default, the unit of striping is a block; however, if the block group factor is specified in order to exploit chunks, the unit of striping is a chunk.

A write affinity depth of 1 indicates that the first copy is written to the writer node. The second copy is written to a different rack. The third copy is written to the same rack as the second copy, but on a different half (which can be composed of several nodes).

A write affinity depth of 2 indicates that the first copy is written to the writer node. The second copy is written to the same rack as the first copy, but on a different half (which can be composed of several nodes). The target node is determined by a hash value on the fileset ID of the file, or it is chosen randomly if the file does not belong to any fileset. The third copy is striped across the disks in a cyclical fashion with the restriction that no two disks are in the same failure group. The following conditions must be met while using a write affinity depth of 2 to get evenly allocated space in all disks:

1. The configuration in disk number, disk size, and node number for each rack must be similar.
2. The number of nodes must be the same in the bottom half and the top half of each rack.

This behavior can be altered on an individual file basis by using the `--write-affinity-failure-group` option of the `mmchattr` command.

This parameter is ignored if write affinity is disabled for the storage pool.

blockGroupFactor=*BlockGroupFactor*
Specifies how many file system blocks are laid out sequentially on disk to behave like a single large block. This option only works if `--allow-write-affinity` is set for the data pool. This applies only to a new data block layout; it does not migrate previously existing data blocks.

See *File Placement Optimizer* in *IBM Spectrum Scale: Administration Guide*.

-a

Specifies asynchronous processing. If this flag is specified, the `mmadddisk` command returns after the file system descriptor is updated and the rebalancing scan is started; it does not wait for rebalancing to finish. If no rebalancing is requested (the `-r` flag not specified), this option has no effect.

-r

Rebalances the file system to improve performance. Rebalancing attempts to distribute file blocks evenly across the disks of the file system. In IBM Spectrum Scale 5.0.0 and later, rebalancing is implemented by a lenient round-robin method that typically runs faster than the previous method of strict round robin. To rebalance the file system using the strict round-robin method, include the `--strict` option that is described in the following text.

--strict
Rebalances the specified files with a strict round-robin method. In IBM Spectrum Scale 4.2.3 and earlier, rebalancing always uses this method.

Note: Rebalancing of files is an I/O intensive and time-consuming operation and is important only for file systems with large files that are mostly invariant. In many cases, normal file update and creation rebalances a file system over time without the cost of a complete rebalancing.

Note: Rebalancing distributes file blocks across all the disks in the cluster that are not suspended, including stopped disks. For stopped disks, rebalancing does not allow read operations and allocates data blocks without writing them to the disk. When the disk is restarted and replicated data is copied onto it, the file system completes the write operations.

-v {yes | no}
Verify that specified disks do not belong to an existing file system. The default is `-v yes`. Specify `-v no` only when you want to reuse disks that are no longer needed for an existing file system. If the command is interrupted for any reason, use the `-v no` option on the next invocation of the command.

Important: Using `-v no` on a disk that already belongs to a file system corrupts that file system. This problem is not detected until the next time that file system is mounted.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies the nodes that are to participate in the restriping of the file system after the specified disks are available for use by GPFS. This parameter must be used with the `-r` option. This command supports all defined node classes. The default is `all` or the current value of the `defaultHelperNodes` parameter of the `mmchconfig` command.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

--qos *QosClass*
Specifies the Quality of Service for I/O operations (QoS) class to which the instance of the command is assigned. If you do not specify this parameter, the instance of the command is assigned by default to the `maintenance` QoS class. This parameter has no effect unless the QoS service is enabled. For more information, see the topic "mmchqos command" on page 263. Specify one of the following QoS classes:

`maintenance`
This QoS class is typically configured to have a smaller share of file system IOPS. Use this class for I/O-intensive, potentially long-running GPFS commands, so that they contribute less to reducing overall file system performance.

other
This QoS class is typically configured to have a larger share of file system IOPS. Use this class for administration commands that are not I/O-intensive.

For more information, see the topic *Setting the Quality of Service for I/O operations (QoS)* in *IBM Spectrum Scale: Administration Guide*.

Exit status

0
Successful completion.

nonzero
A failure occurred.

Security

You must have root authority to run the `mmadddisk` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. Assume that the file `./newNSDstanza` contains the following NSD stanza:

```
%nsd: nsd=gpfs10nsd
  servers=k148n07,k148n06
  usage=dataOnly
  failureGroup=5
  pool=pool2
  thinDiskType=nvme
```

   To add the disk that is defined in this stanza, run the following command:

```
mmadddisk fs1 -F ./newNSDstanza -r
```

   The command displays information like the following example:

```
GPFS: 6027-531 The following disks of fs1 will be formatted on node
 k148n07.kgn.ibm.com:
    gpfs10nsd: size 2202 MB
Extending Allocation Map
Creating Allocation Map for storage pool 'pool2'
  75 % complete on Thu Feb 16 13:57:52 2006
 100 % complete on Thu Feb 16 13:57:54 2006
Flushing Allocation Map for storage pool 'pool2'
GPFS: 6027-535 Disks up to size 24 GB can be added to storage
pool pool2.
Checking allocation map for storage pool system
  62 % complete on Thu Feb 16 13:58:03 2006
 100 % complete on Thu Feb 16 13:58:06 2006
Checking allocation map for storage pool pool1
  62 % complete on Thu Feb 16 13:58:11 2006
 100 % complete on Thu Feb 16 13:58:14 2006
Checking allocation map for storage pool pool2
  63 % complete on Thu Feb 16 13:58:19 2006
 100 % complete on Thu Feb 16 13:58:22 2006
GPFS: 6027-1503 Completed adding disks to file system fs1.
mmadddisk: 6027-1371 Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
Restriping fs1 ...
GPFS: 6027-589 Scanning file system metadata, phase 1 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 2 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 3 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 4 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-565 Scanning user file metadata ...
  68 %  complete on Thu Feb 16 13:59:06 2006
 100 %  complete on Thu Feb 16 13:59:07 2006
GPFS: 6027-552 Scan completed successfully.
Done
```

See also

- “mmchdisk command” on page 212
- “mmcrnsd command” on page 335
- “mmdeldisk command” on page 364
- “mmlsdisk command” on page 497
- “mmlsnsd command” on page 522
- “mmlspool command” on page 528

Location

`/usr/lpp/mmfs/bin`

mmaddnode command

Adds nodes to a GPFS cluster.

Synopsis

```
mmaddnode -N {NodeDesc[,NodeDesc...] | NodeFile} [--accept] [--on-error-continue]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the mmaddnode command to add nodes to an existing IBM Spectrum Scale cluster. On each new node, a mount point directory and character mode device is created for each GPFS file system.

Follow these rules when adding nodes to an IBM Spectrum Scale cluster:

- You may issue the command only from a node that already belongs to the IBM Spectrum Scale cluster.
- IBM Spectrum Scale must be installed on the nodes before you run the **mmaddnode** command to add those nodes to the cluster.
- While a node may mount file systems from multiple clusters, the node itself may only be added to a single cluster using the mmcrcluster or mmaddnode command.
- The nodes must be available for the command to be successful. If any of the nodes listed are not available when the command is issued, a message listing those nodes is displayed. You must correct the problem on each node and reissue the command to add those nodes.
- After the nodes are added to the cluster, use the mmchlicense command to designate appropriate IBM Spectrum Scale licenses to the new nodes.

Parameters

-N *NodeDesc*[,*NodeDesc*...] | *NodeFile*
Specifies node descriptors, which provide information about nodes to be added to the cluster.

NodeFile
Specifies a file containing a list of node descriptors, one per line, to be added to the cluster.

***NodeDesc*[,*NodeDesc*...]**
Specifies the list of nodes and node designations to be added to the IBM Spectrum Scale cluster. Node descriptors are defined as:

```
NodeName:NodeDesignations:AdminNodeName:LicenseType
```

where:

NodeName
Specifies the host name or IP address of the node for GPFS daemon-to-daemon communication.

The host name or IP address must refer to the communication adapter over which the GPFS daemons communicate. Aliased interfaces are not allowed. Use the original address or a name that is resolved by the **host** command to that original address. You can specify a node using any of these forms:

- Short host name (for example, h135n01)
- Long, fully-qualified, host name (for example, h135n01.ibm.com)

- IP address (for example, 7.111.12.102). IPv6 addresses must be enclosed in brackets (for example, [2001:192::192:168:115:124]).

Regardless of which form you use, GPFS will resolve the input to a host name and an IP address and will store these in its configuration files. It is expected that those values will not change while the node belongs to the cluster.

NodeDesignations
An optional, "-" separated list of node roles:

- **manager** | **client** – Indicates whether a node is part of the node pool from which file system managers and token managers can be selected. The default is **client**.
- **quorum** | **nonquorum** – Indicates whether a node is counted as a quorum node. The default is **nonquorum**.

  Note: If you are designating a new node as a quorum node, and `adminMode central` is in effect for the cluster, GPFS must be down on all nodes in the cluster. Alternatively, you may choose to add the new nodes as `nonquorum` and once GPFS has been successfully started on the new nodes, you can change their designation to `quorum` using the `mmchnode` command.
- **perfmon** - Designates a node as a performance monitoring (*perfmon*) node. The *perfmon* designation is ignored if the node is not a Linux node or if the performance monitoring configuration has not been setup yet. For more details about how to generate the configuration of the performance monitoring tool, see *mmperfmon command* in *IBM Spectrum Scale: Command and Programming Reference*.

AdminNodeName
Specifies an optional field that consists of a node interface name to be used by the administration commands to communicate between nodes. If *AdminNodeName* is not specified, the *NodeName* value is used.

Note: *AdminNodeName* must be a resolvable network host name. For more information, see the topic *GPFS node adapter interface names* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

LicenseType
Assigns a license of the specified type to the node. Valid values are `server`, `client`, and `fpo`. For information about these license types, see "mmchlicense command" on page 239.

The full text of the Licensing Agreement is provided with the installation media and can be found at the IBM Software license agreements website (www.ibm.com/software/sla/sladb.nsf).

`--accept`
Specifies that you accept the terms of the applicable license agreement. If one or more node descriptors includes a *LicenseType* term, this parameter causes the command to skip the prompt for you to accept a license.

`--on-error-continue`
Enables the command to trap invalid nodes, remove those nodes from the node list, and continue to process the command. Without selecting this option, the command may exit for any reason when a node is found to be invalid.

Note: Not all error conditions relating to a node or the program function can allow the command to continue.

When this option is used, the *add node* process runs in two stages and the progress of each stage is displayed on the console as it runs. The `--on-error-continue` option provides the following functions:

1. Some errors are trapped and the nodes that generated the errors are removed from the list of valid nodes to process. This allows the program to continue as long as there is at least one valid node remaining in the *add node* process.

2. Presents the console output in more of a report format.
3. Creates two stages in the *add node* process. The first stage is where some node errors can be trapped and thus the node gets excluded from the process. The second stage is more critical and will error out on any significant errors.
4. In stage one, if nodes failed to get added to the cluster, two files are created to list the valid nodes and invalid nodes separately. The user gets informed of these files. This allows the user to rerun with just the valid node list.

Refer to the *Example* section for the example of the *add node* process with the `--on-error-continue` option enabled.

You must provide a *NodeDesc* for each node to be added to the IBM Spectrum Scale cluster.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmaddnode` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see the topic *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

Examples

1. To add nodes k164n06 and k164n07 as `quorum` nodes, designating k164n06 to be available as a **manager** node, issue the following command:

```
# mmaddnode -N k164n06:quorum-manager,k164n07:quorum
```

To confirm the addition, issue the following command:

```
# mmlscluster
```

A sample output is as follows:

```
GPFS cluster information
========================
  GPFS cluster name:       cluster1.kgn.ibm.com
  GPFS cluster id:         680681562214606028
  GPFS UID domain:         cluster1.kgn.ibm.com
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           server-based

GPFS cluster configuration servers:
-----------------------------------
  Primary server:    k164n07.kgn.ibm.com
  Secondary server:  k164n04.kgn.ibm.com

Node Daemon node name      IP address    Admin node name  Designation
-----------------------------------------------------------------------
 1   k164n04.kgn.ibm.com   198.117.68.68  k164n04.kgn.ibm.com  quorum
 2   k164n07.kgn.ibm.com   198.117.68.71  k164n07.kgn.ibm.com  quorum
 3   k164n06.kgn.ibm.com   198.117.68.70  k164n06.kgn.ibm.com  quorum-manager
```

2. In the following example the mmaddnode command adds a node without specifying a license:

```
(11:37:06) c34f2n03:~ # mmaddnode -N c6f2bc4n8:quorum
Tue Mar 12 11:37:13 EDT 2019: mmaddnode: Processing node c6f2bc4n8.gpfs.net
mmaddnode: Command successfully completed
mmaddnode: Warning: Not all nodes have proper GPFS license designations.
    Use the mmchlicense command to designate licenses as needed.
mmaddnode: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

   The command displays a warning message that some nodes do not have licenses.

3. In the following example the mmaddnode command specifies a license to be designated to the node:

```
 # mmaddnode -N c6f2bc4n8:quorum::server
mmaddnode: Node c6f2bc4n8.gpfs.net will be designated as possessing server license.
Please confirm that you accept the terms of the IBM Spectrum Scale server Licensing
Agreement.
The full text can be found at www.ibm.com/software/sla
Enter "yes" or "no": yes
Tue Mar 12 11:41:51 EDT 2019: mmaddnode: Processing node c6f2bc4n8.gpfs.net
mmaddnode: Command successfully completed
mmaddnode: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

   The command prompts the user to accept the terms of the licensing agreement.

4. In the following example the mmaddnode command specifies a license and also specifies the `--accept` parameter:

```
 # mmaddnode -N c6f2bc4n8:quorum::server --accept
mmaddnode: Node c6f2bc4n8.gpfs.net will be designated as possessing server license.
Tue Mar 12 11:51:15 EDT 2019: mmaddnode: Processing node c6f2bc4n8.gpfs.net
mmaddnode: Command successfully completed
mmtrace: move /tmp/mmfs/lxtrace.trc.c34f2n03.cpu0 /tmp/mmfs/
trcfile.2019-03-12_11.51.27.28959.c34f2n03.cpu0
mmtrace: formatting /tmp/mmfs/trcfile.2019-03-12_11.51.27.28959.c34f2n03 to /tmp/mmfs/
trcrpt.2019-03-12_11.51.27.28959.c34f2n03.gz
mmaddnode: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

   Because the `--accept` parameter is specified, the *command* does not need to prompt for acceptance of the license agreement.

5. In the following example, the **mmaddnode** command is run with the `--on-error-continue` option. This option helps to continue the process of adding a list of nodes to the cluster even if certain nodes in the list fail to get added to the cluster.

```
# mmaddnode -N /tmp/input_nodes.in  --on-error-continue
mmaddnode: On Error Continue option detected

=====================================================================
                          Stage 1
Running new node validation and remote initial config...
=====================================================================
---------------------------------------------------------------------
PROCESSING NODE: 1
mmaddnode: Node c71f2u33p1.gpfs.net will be designated as possessing server license.
Tue Feb 16 01:31:50 EST 2021: mmaddnode: Processing node c71f2u33p1.gpfs.net
---------------------------------------------------------------------
PROCESSING NODE: 2
mmaddnode: Incorrect node c71f2u33p1.gpfs.net1 specified for command.
---------------------------------------------------------------------
PROCESSING NODE: 3
mmaddnode: Incorrect node c71f2u33p1.gpfs.net2 specified for command.
---------------------------------------------------------------------
PROCESSING NODE: 4
mmaddnode: Incorrect node c71f2u33p1.gpfs.net3 specified for command.
---------------------------------------------------------------------
PROCESSING NODE: 5
mmaddnode: Node c21m2n04.gpfs.net already belongs to the GPFS cluster.
---------------------------------------------------------------------
=====================================================================
                        Stage 1 Complete
Number of valid nodes: 1
Total number of nodes: 5
```

```
=============================================================

Stage 1 generated files:
      /tmp/mmaddnode_rpt__2021_02_16_01_31_47/valid_nodes_list
      /tmp/mmaddnode_rpt__2021_02_16_01_31_47/invalid_nodes_list

      If stage 2 fails, address any concerns, then rerun the command
      using the 'valid_nodes_list' file with the -N option to avoid
      processing invalid nodes again.
      Rerun the command with the invalid_nodes_list file with the -N
      option once you have addressed the concerns with those invalid
      nodes.

=============================================================
                          Stage 2
Running cluster checks and config repository updates.
This stage has critical checks.  It does not generate valid/invalid
node reports and will exit on any serious errors.
=============================================================
=============================================================
                       Stage 2 Complete
=============================================================
mmaddnode: Command successfully completed
mmaddnode: Propagating the cluster configuration data to all affected nodes.
```

See also

- “mmchconfig command” on page 170
- “mmcrcluster command” on page 306
- “mmchcluster command” on page 165
- “mmdelnode command” on page 375
- “mmlscluster command” on page 492

Location

/usr/lpp/mmfs/bin

mmadquery command

Queries and validates Active Directory (AD) server settings.

Synopsis

```
mmadquery list {user | uids | gids | groups | dc | trusts | idrange} [Options]
```

or

```
mmadquery check {uids | gids | idrange} [Options]
```

or

```
mmadquery stats {user |uids}
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmadquery` command to query an AD Server for users, groups, user IDs, group IDs, known domain controller and trusts, and to run consistency checks.

Parameters

user
Queries and lists the defined users.

uids
Queries and lists the defined users with user IDs and group IDs.

gids
Queries and lists the defined groups with group IDs.

groups
Queries and lists the defined groups.

dc
Queries and lists the defined domain controllers.

trusts
Queries and lists the defined trusts.

idrange
Queries and lists the ID range used by a given AD server.

Options

--server *SERVER*
Specifies the IP address of the AD server you want to query. If you do not specify a server, `mmadquery` attempts to get the AD server from the `/etc/resolv.conf` file (nameserver).

Note: This option should be used along with the `domain` option, which is provided in the following section.

--domain *DOMAIN*
Specifies the Windows domain. If you do not specify a domain, `mmadquery` uses `nslookup` to determine the domain based on the server.

Note: This option should be used along with the `server` option.

--user *USER*
Specifies the AD user used to run the LDAP query against the AD server. The default is Administrator.

--pwd-file *File*
Specifies the file that contains a password to use for authentication.

--filter *FILTER*
Specifies a search phrase to limit the number of LDAP objects, thus is applied only to first column of output. Every LDAP object beginning with the search phrase is queried.

--CSV
Shows output in machine parseable (CSV) format.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

--debug or -d
Shows debugging information

--basedn or -b
Includes basedn for LDAP objects queried in query output. This option is not supported when querying idrange or running a 'stats' query.

--traverse
Traverses all known domains and provide query output for all domains that are detected.

--long or -L
Indicates that you want to see more details. For more information, see Level of query detail below. This option is not supported for the "stats" queries.

Level of query detail

Table 10. Query details by type

Query	Additional content
User	Group membership
DC	Operating system
UIDs	GID, Primary Group ID
Trusts	DC

Exit status

0
No errors found.

1
No arguments specified.

10
Failed a check.

11
Unable to determine the AD server to check.

12
Unable to determine the domain.

13
Failed to construct a basedn for an LDAP query.

99
Access to the AD server failed, can be incorrect password, user, or domain.

Security

You must have root authority to run the mmadquery command. For more information, see the topic *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

Examples

1. To show a list of users for the AD server, run the following command:

```
# mmadquery list user --pwd-file /tmp/mmadquery.cfg
```

A sample output is as follows:

```
USER from server 9.155.106.234 (domain subdom1.mzdom.com)
          User
 -------------
  Administrator
          Guest
         krbtgt
         MZDOM$
        aduser1
        aduser2
        Taduser3
```

2. To show a list of groups for the AD server, run the following command:

```
# mmadquey list groups --pwd-file /tmp/mmadquery.cfg
```

A sample output is as follows:

```
GROUPS from server 9.155.106.234 (domain subdom1.mzdom.com)
                                        Group
      ---------------------------------------
                             Domain Computers
                              Cert Publishers
                                 Domain Users
                                Domain Guests
                           RAS and IAS Servers
                                 Domain Admins
                                 Schema Admins
                              Enterprise Admins
                  Group Policy Creator Owners
      Allowed RODC Password Replication Group
       Denied RODC Password Replication Group
      Enterprise Read-only Domain Controllers
                             Domain Controllers
                Read-only Domain Controllers
                                     DnsAdmins
                                DnsUpdateProxy
                                       UNIXGRP
                                unmapped group
                                          bla
```

3. To check user IDs against locally defined ID mapping range, issue the following command:

```
# mmadquery check uids --pwd-file /tmp/mmadquery.cfg -L
```

A sample output is as follows:

```
UIDS from server 9.155.106.234 (domain subdom1.mzdom.com)
        User   SID                                          UID  UIDNumber GIDNumber Primary Group ID
 ------------------------------------------------------------------------------------------------
Guest          S-1-5-21-2808815044-4164012579-2832416960-501  -   -         -           514
SUBDOM1$       S-1-5-21-2808815044-4164012579-2832416960-1103 -   -         -           513
Administrator  S-1-5-21-2808815044-4164012579-2832416960-500  -   -         -           513
```

```
       krbtgt  S-1-5-21-2808815044-4164012579-2832416960-502  -   -         -          513
       User 1  S-1-5-21-2808815044-4164012579-2832416960-1107 -   -         -          513
      aduser1  S-1-5-21-2808815044-4164012579-2832416960-1601 aduser1 20000007 20000008 513
       User 2  S-1-5-21-2808815044-4164012579-2832416960-1110 aduser 10001     20000009 513
WARNING: UID of user User 2 outside id mapping range 'mzdom'.
```

4. To show a list of users with group membership by domain, run the following command:

```
# mmadquery list user -L --pwd-file /tmp/mmadquery.cfg --traverse
```

A sample output is as follows:

```
USER from server 9.155.106.232 (domain mzdom.com)
        User                                                                                    Groups
        ------------------------------------------------------------------------------------    ---------
        Guest
Guests

        SUBDOM1$
Administrator Group Policy Creator Owners,Enterprise Admins,Schema Admins,Domain Admins,Administrators
        krbtgt                                                  Denied RODC Password Replication
Group
        aduser1                                                                                     Administrators
        aduser2                                                                                 bla,unmapped
group
        aduser3
        aduser4

USER from server 9.155.106.234 (domain subdom1.mzdom.com)
        User                                            Groups
        ------------ ------------------------------------------------------------
        Administrator Group Policy Creator Owners,Domain Admins,Administrators
        Guest
Guests
        krbtgt            Denied RODC Password Replication Group Administrators
        MZDOM$
        aduser1
        aduser2
        aduser3
        aduser4
```

5. To show the number of users by group and domain, run the following command:

```
# mmadquery stats user -L --pwd-file /tmp/mmadquery.cfg --traverse
```

A sample output is as follows:

```
USER from server 9.155.106.232 (domain mzdom.com)
                                        Group Count
        ------------------------------------ -----
                                         TOTAL     7
                                        Guests     1
                    Group Policy Creator Owners     1
                              Enterprise Admins     1
                                  Schema Admins     1
                                  Domain Admins     1
                                 Administrators     2
           Denied RODC Password Replication Group  1
                                           bla     1
                                 unmapped group     1
USER from server 198.51.100.13 (domain subdom1.mzdom.com)
                                        Group Count
        ------------------------------------ -----
                                         TOTAL     7
                    Group Policy Creator Owners     1
                                  Domain Admins     1
                                 Administrators     2
                                        Guests     1
        Denied RODC Password Replication Group     1
```

6. To show a list of the number of unmapped users, run the following command:

```
# mmadquery stats uids --pwd-file /tmp/mmadquery.cfg
```

A sample output is as follows:

```
UIDS from server 9.155.106.232 (domain mzdom.com)
Group      Count
---------  -----
TOTAL      7
```

```
MAPPED     2
UN-MAPPED  5
```

7. To check group IDs against locally defined ID map, run the following command:

```
# mmadquery check gids -L --pwd-file /tmp/mmadquery.cfg
```

A sample output is as follows:

GIDS from server 9.155.106.232 (domain w2k8r2-dom02.mzdom.com)

```
GIDS from server 9.155.106.232 (domain w2k8r2-dom02.mzdom.com)
                              Group                                                SID UID  UIDNumber          GIDNumber
 ------------------------------------------------------------------------------------------------------------------------
                       Domain Computers S-1-5-21-2808815044-4164012579-2832416960-515      -          -                    -
                        Cert Publishers S-1-5-21-2808815044-4164012579-2832416960-517      -          -                    -
                           Domain Users S-1-5-21-2808815044-4164012579-2832416960-513      -          -  20000008         -
                          Domain Guests S-1-5-21-2808815044-4164012579-2832416960-514      -          -                    -
                     RAS and IAS Servers S-1-5-21-2808815044-4164012579-2832416960-553      -          -                    -
                     Domain Admins       S-1-5-21-2808815044-4164012579-2832416960-512      -          -                    -
                     Schema Admins       S-1-5-21-2808815044-4164012579-2832416960-518      -          -                    -
                     Enterprise Admins   S-1-5-21-2808815044-4164012579-2832416960-519      -          -                    -
               Group Policy Creator Owners S-1-5-21-2808815044-4164012579-2832416960-520      -          -                    -
Allowed RODC Password Replication Group S-1-5-21-2808815044-4164012579-2832416960-571      -          -                    -
Denied RODC Password Replication Group  S-1-5-21-2808815044-4164012579-2832416960-57       -          -                    -
Enterprise Read-only Domain Controllers S-1-5-21-2808815044-4164012579-2832416960-498      -          -                    -
                     Domain Controllers S-1-5-21-2808815044-4164012579-2832416960-516      -          -                    -
Read-only Domain Controllers   S-1-5-21-2808815044-4164012579-2832416960-521              -          -                    -
                   DnsAdmins   S-1-5-21-2808815044-4164012579-2832416960-1101             -          -                    -
              DnsUpdateProxy   S-1-5-21-2808815044-4164012579-2832416960-1102             -          -                    -
                     UNIXGRP   S-1-5-21-2808815044-4164012579-2832416960-1104             -          - 200002222          -
              unmapped group   S-1-5-21-2808815044-4164012579-2832416960-1603             -          -                    -
                       bla     S-1-5-21-2808815044-4164012579-2832416960-1604             -          -                    -
-WARNING: GID of group 'UNIXGRP' outside id mapping range 'mzdom'.
```

8. To show a list of domain controllers, run the following command:

```
# mmadquery list dc L --pwd-file /tmp/mmadquery.cfg
```

A sample output is as follows:

```
DC from server 9.155.106.232 (domain w2k8r2-dom02.mzdom.com)
      DC              Hostname               Operating System
------------- ---------------------  ------------------------------
WW2K8R2-DOM03 w2k8r2-dom03.mzdom.com Windows Server 2008 R2 Standard
WW2K8R2-DOM02 w2k8r2-dom02.mzdom.com Windows Server 2008 R2 Standard
```

9. To show a list of trusts, run the following command:

```
# mmadquery list trusts --pwd-file /tmp/mmadquery.cfg
```

A sample output is as follows:

```
TRUSTS from server 9.155.106.232 (domain w2k8r2-dom02.mzdom.com)
       DC                        Trust Type
-----------------     ----------------------------
subdom1.mzdom.com     Within Forest bi-directional
     w2k12dom.com       Forest Transitive outbound
```

10. To show a list of ID ranges and to check whether any IDs on the Ad server are outside of the locally defined ID range, run the following command:

```
# mmadquery check idrange --pwd-file /tmp/mmadquery.cfg
```

A sample output is as follows:

```
IDRANGE from server 9.155.106.232 (domain w2k8r2-dom02.mzdom.com)
 Domain       IDRange        IDMapRange
--------- --------------- -----------------
msdom.com 10001-200000000 20000000-25999999
WARNING: IDs from domain 'mzdom.com' are outside locally defined id mapping range 'mzdom'.
```

11. To show a list of ID ranges by domain, run the following command:

```
# mmadquery list idrange --pwd-file /tmp/mmadquery.cfg -L --traverse
```

A sample output is as follows:

```
IDRANGE from server 9.155.106.232 (domain mzdom.com)
   Domain               IDRange        IDMapRange
---------------- ----------------- ----------------
mzdom.com         10001--260000009   10000000-29999999


IDRANGE from server 9.155.106.234 (domain subdom1.mzdom.com)

   Domain               IDRange        IDMapRange
---------------- ----------------- ----------------
subdom1.mzdom.com 200000001-26000010 10000000-29999999
```

Location

/usr/lpp/mmfs/bin

mmafmconfig command

Can be used to manage home caching behavior and mapping of gateways and home NFS exported servers.

Synopsis

You can use the **mmafmconfig** command to -

- Set up or update mapping for parallel data transfers by using add, update, or delete options.
- Enable or disable extended attributes/sparse file support from the AFM cache.

```
mmafmconfig {add | update} MapName --export-map ExportServerMap [--no-server-resolution]
```

or

```
mmafmconfig delete {MapName | all}
```

or

```
mmafmconfig show [-Y] [MapName | all]
```

or

```
mmafmconfig {enable | disable} ExportPath
```

Availability

Available on all IBM Spectrum Scale editions. Available on AIX and Linux.

Description

You can use this command on the home cluster to enable support of extended attributes, sparse files on an AFM cache fileset pointing to this home. You must run the **mmafmconfig enable** command on the home export or target path. Running this command creates the `.afm` directory that contains the control-enabled, directio `'.afmctl'` file. The **mmafmconfig disable** command removes the `.afm` directory from the home export or target path and subsequently, the cache does not support synchronization of sparse files and files with extended attributes.

You can also use the **mmafmconfig** command with add, update, delete, or show options on the cache site to manage mapping of gateway node with home NFS servers for parallel data transfers.

You must run the **mmafmconfig enable** command at a home fileset path before you link the AFM fileset at the cache site.

If the AFM cache fileset was linked without running the **mmafmconfig enable** command at the home export path, you need to issue the following commands:

1. Issue the following command at the home:

   ```
   # mmafmconfig enable HOME_EXPORT_PATH
   ```

2. Issue the following commands at the cache:

   ```
   # mmafmctl FS stop -j AFM_Fileset
   ```

   ```
   # mmafmctl FS start -j AFM_Fileset
   ```

Parameters

MapName
Specifies the name that uniquely identifies the mapping of the gateway nodes with the home NFS exported servers.

--export-map *ExportServerMap*
Specifies a comma-separated list of pairs of home NFS exported server nodes (*ExportServer*) and gateway nodes (*GatewayNode*), in the following format:

```
[ExportServer/GatewayNode][,ExportServer/GatewayNode][,...]
```

where:

ExportServer
Is the IP address or host name of a member node in the home cluster *MapName*.

GatewayNode
Specifies a gateway node in the cache cluster (the cluster where the command is issued).

--no-server-resolution
When this option is specified, AFM skips the DNS resolution of a specified hostname and creates the mapping. The specified hostname in the mapping is not be replaced with an IP address. Ensure that the specified hostname is resolvable while the mapping is in operation.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

enable
Enables extended attributes or sparse files functions on the AFM cache. Run at the home cluster only.

disable
Disables extended attributes or sparse files functions on the AFM cache. Run at the home cluster only.

ExportPath
Specifies the root of the home exported directory for enabling or disabling the AFM features.

add
Sets up maps for parallel data transfers. Run at cache only.

delete
Deletes maps for parallel data transfers. Run at cache only.

update
Updates maps for parallel data transfers. Run at cache only.

show
Displays all the existing maps. Each map displays the mapping of a gateway node and home NFS server pair that is separated by a comma ','.

Exit status

0
Successful completion.

Nonzero
A failure has occurred.

Security

You must have root authority to run the `mmafmconfig` command.

The node on which the command is issued must be able to run remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see the topic *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

Example

The following is an example of a mapping for NFS targets, adding all gateway nodes and NFS servers, and using these mapping for creating AFM filesets. In this example, hs22n18, hs22n19, hs22n20, and hs22n21 nodes are assumed as cache gateway nodes. The gateway nodes are mapped to two home NFS servers js22n01 with an IP address 192.168.200.11 and js22n02 with an IP address 192.168.200.12.

1. Issue the following command:

```
# mmafmconfig add mapping1 --export-map js22n01/hs22n18,js22n02/hs22n19,
js22n01/hs22n20,js22n02/hs22n21
```

   The system displays output similar to:
   ```
   mmafmconfig: Command successfully completed.
   ```

2. Issue the following command:

```
# mmafmconfig show
```

   The system displays output similar to:
   ```
   Map name: mapping1
   Export server map: 192.168.200.11/hs22n18.gpfs.net,192.168.200.12/
   hs22n19.gpfs.net,92.168.200.11/hs22n20.gpfs.net,192.168.200.12/
   hs22n21.gpfs.net
   ```

3. Issue the following commands to create filesets by using *mapping1*:

```
# mmcrfileset gpfs1 sw1 -p afmmode=sw,afmtarget=nfs://mapping1/gpfs/gpfs2/swhome --inode-
space new
```

```
# mmcrfileset gpfs1 ro1 -p afmmode=ro,afmtarget=nfs://mapping1/gpfs/gpfs2/swhome --inode-
space new
```

See also

- “mmafmctl command” on page 61
- “mmafmlocal command” on page 78
- “mmchconfig command” on page 170
- “mmchfileset command” on page 224
- “mmchfs command” on page 232
- “mmcrfileset command” on page 311
- “mmcrfs command” on page 318
- “mmlsconfig command” on page 495
- “mmlsfileset command” on page 501
- “mmlsfs command” on page 506

Location

/usr/lpp/mmfs/bin

mmafmcosaccess command

Maps a directory in an AFM to cloud object storage fileset to the bucket on a cloud object storage.

Synopsis

```
mmafmcosaccess Device FilesetName Path {set | get | delete}
               [--bucket bucket --endpoint endpoint {--akey AccessKey
               --skey SecretKey | --keyfile filePath}]
```

or

```
mmafmcosaccess Device FilesetName Path get --report
```

Availability

Available on all IBM Spectrum Scale editions.

Description

This command is used to map individual directories to a bucket on the cloud object storage.

Parameters

Device
Specifies a device name of a file system to contain a new fileset.

FilesetName
Specifies a name of the AFM to cloud object storage enabled fileset that has an existing relationship.

Path
Specifies a path of a directory under an AFM to cloud object storage fileset that needs to be mapped to a cloud object storage.

set
Stores the credentials and maps a directory and a bucket.

get
Retrieves the stored credentials that were set for a directory. AFM shows an error if the bucket and server name combination does not exist.

delete
Deletes the mapping of a directory and a bucket.

--akey *AccessKey*
Specifies an access key that is used for communication with a cloud object storage.

--skey *SecretKey*
Specifies a secret key that is used for communication with a cloud object storage.

--bucket
Specifies a unique bucket on a cloud object storage. AFM will use this bucket as a target for a directory and it will synchronize the data between the bucket and the directory by using the stored credentials.

--endpoint
Specifies an endpoint that is the address of a cloud object storage server on which it receives requests from clients. An endpoint can be either HTTP or HTTPS. It can either be a DNS qualified hostname or an IP address. An endpoint also contains a port number on which a cloud object storage server is running.

--keyfile
Specifies a key file that contains an access key and a secret key. Instead of providing the access key and the secret key on the command line, a key file can be used. The key file must contain two lines for

akey and skey separated by a colon. An example of the format of a key file */root/keyfile1* is as follows:

```
akey:AccessKey
skey:SecretKey
```

--report
This option must be used with the `get` option. It generates a report of a specified directory that is mapped to a cloud object storage.

Note: If access and secret keys are stored by using the **mmafmcoskeys** command, you need not to use the `--akey` or `--skey` or `--keyfile` options with this command.

Exit status

0
Successful completion.

nonzero
A failure occurred.

Security

The node on which this command is issued must be able to run remote shell commands on any other node in the cluster. The node must run these remote shell commands without a password and must not produce any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

Examples

1. To map a directory inside an AFM to cloud object storage fileset, issue the following command:

```
# mmafmcosaccess fs1 SW1 /gpfs/fs1/SW1/dir1 set --bucket vault1 --endpoint http://IP:port --keyfile /root/
keyfile1
```

2. To retrieve a directory that is mapped inside an AFM to cloud object storage fileset, issue the following command:

```
# mmafmcosaccess fs1 SW1 /gpfs/fs1/SW1/dir1 get
```

3. To delete a directory that is mapped inside an AFM to cloud object storage fileset, issue the following command:

```
# mmafmcosaccess fs1 SW1 /gpfs/fs1/SW1/dir1 delete
```

4. To generate a report of a mapped directory, issue the following command:

```
# mmafmcosaccess fs1 SW1 /gpfs/fs1/SW1/dir1 get --report
```

See also

- “mmafmcosconfig command” on page 50
- “mmafmcosctl command” on page 55
- “mmafmcoskeys command” on page 58

Location

`/usr/lpp/mmfs/bin`

mmafmcosconfig command

Creates and displays an AFM to cloud object storage fileset.

Synopsis

```
mmafmcosconfig Device FilesetName --endpoint http[s]://{[Region@]Server |
               [Region@]ExportMap}[:port]
               [--object-fs --cleanup --xattr --ssl-cert-verify --user-keys
               {--bucket BucketName | --new-bucket BucketName} --dir Path
               {--policy PolicyFile | --tmpdir DirectoryPattern --tmpfile FilePattern}
                --quota-files NumberOfFiles --quota-blocks NumberOfBlocks
                --uid UID --gid GID --perm Permission --mode AccessMode
                --chunk-size Size --read-size Size
                --acls {--vhb | --gcs} ]
```

or

```
mmafmcosconfig Device FilesetName --report
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The **mmamfcosconfig** command constructs a new AFM to cloud object storage fileset with the specified name parameters. A single command is provided to create and link the AFM to cloud object storage fileset. You can specify the parameters such as policies, mode, and permissions to save running extra commands. To modify any parameters after the creation of the AFM to cloud object storage fileset, you need to run a specific command separately such as establishing policy or modifying quota on the fileset.

A report that has all details of the AFM to cloud object storage fileset can be generated.

Before you run this command, ensure that the `gpfs.afm.cos` package is installed on the gateway nodes.

For more information about filesets, see the *filesets* section in the *IBM Spectrum Scale: Administration Guide*.

Parameters

Device
Specifies a device name of a file system to contain a new fileset.

FilesetName
Specifies a name of an AFM to cloud object storage fileset to be created.

Note: The following restrictions should be applied on the fileset names:

- The name must be unique within the file system.
- The length of the name must be in the range 1-255.
- The name root is reserved for a fileset of the root directory of the file system.
- The name cannot be the reserved word "new". However, the character string "new" can appear within a fileset name.
- The name cannot begin with a hyphen (-).
- The name cannot contain these characters: / ? $ & * ( ) ` # | [ ] \fR.
- The name cannot contain a white-space character such as blank space or tab.

--endpoint
Specifies an endpoint that is the address of a cloud object storage server on which it receives requests from clients. An endpoint can be either HTTP or HTTPS. It can also be a DNS qualified hostname or an IP address. An endpoint also contains a port number on which the cloud object storage server is running.

Region
Specifies a region where a bucket is located on a cloud object server. You can specify a region with a server or a map name by separating them with '@'.

port
Specifies a port number on a cloud object storage server on which it is listening from the clients. Default port is 80.

--object-fs
Specifies the behavior of an AFM to cloud object storage fileset. The following two modes can be enabled on an AFM to cloud object storage fileset:

ObjectFS
This behavior is used to enable auto-synchronization of metadata from a cloud object storage server to the AFM to cloud object storage enabled fileset. This mode allows AFM to auto-synchronization fileset metadata when a lookup or readdir operation is performed on the fileset or the AFM refresh intervals are triggered.

AFM downloads information of objects from the cloud object storage automatically and presents it as files on the AFM to cloud object storage fileset. The ObjectFS enabled AFM to cloud object storage fileset behaves in a similar way as an AFM mode fileset on-demand behavior.

For the AFM RO, LU, and IW mode enabled AFM to cloud object storage fileset, AFM automatically synchronizes objects from the cloud object storage to the files on the cache. Modification of objects on cloud object storage will be refreshed on the cache.

For the SW and IW mode enabled AFM to cloud object storage fileset, modification on the cache will queue an upload operation to be synchronized as an object to the cloud storage server.

Enable this mode if you want AFM to cloud object storage fileset behave like an AFM mode fileset. This behavior generates traffic between the cloud object storage server and the AFM to cloud object storage fileset.

ObjectOnly
If the **`--object-fs`** parameter is not defined, the ObjectOnly mode will be set. This is the default behavior, which is set on the AFM to cloud object storage server.

With the ObjectOnly mode, refresh of metadata from the cloud object server to an AFM to cloud object storage fileset will not be on-demand or frequent. You need to manually download data or metadata from the cloud object storage to the AFM to cloud object storage fileset.

Meanwhile data synchronization from the AFM to cloud object storage (SW, IW mode) to the cloud object storage will work automatically without manual intervention. This behavior is similar to the AFM fileset mode behavior.

Enable this mode on an AFM to cloud object storage fileset to avoid frequent trips and reduce the network contention by selectively performing the operations.

--cleanup
Deletes and cleans up an existing fileset with the same name along with data and re-creates a new AFM to cloud object storage fileset with given parameters.

Note: Use this option only if you want to delete an existing fileset with the same name and along with data.

--xattr
Specifies user-extended attributes to be synchronized with a cloud object storage. If this option is skipped, the AFM to cloud object storage does not synchronize user-extended attributes with the cloud object storage.

--ssl-cert-verify
Specifies SSL certificate verification. This option is valid only with HTTPS. Default value of this parameter is disabled.

--user-keys
Specifies adding a callback to collect an access key and a secret key. To collect all the information, this option checks the `/var/mmfs/etc/mmuid2keys` file.

The callback reads the `mmuid2keys` file and collects the information. Therefore, you need not to provide access or secret keys manually. An example of a `mmuid2keys` file is as follows:

```
#!/usr/lpp/mmfs/bin/mmksh
# input:
#
#- UID
#- bucket
#- endpoint
#
# output:
#
#- accesskey:secret
#
cmd=$1
uid=$2
bucket=$3
endpoint=$4
keys="akey:skey"
line=$(egrep $uid: /etc/passwd)
nuid=${line%%:*}
exit 0
```

--bucket
Identifies a unique bucket on a cloud object storage. AFM will use this bucket as the target for an AFM to cloud object storage fileset, and it will synchronize data between the AFM to cloud object storage fileset and the bucket on the cloud object storage.

Credentials to access this bucket must be already set by using the **mmafmcoskeys** command for this bucket.

--new-bucket
Specifies a new bucket to be created on a cloud object storage by the AFM to cloud object storage and used this as the Target.

The name of the bucket is as per the support of the cloud object storage. Check the cloud object storage for bucket name guidelines.

Credentials must be already set by using the **mmafmcoskeys** command to create this bucket on the cloud object storage.

--dir
Specifies a junction path to link an AFM to cloud object storage fileset inside a file system. If not specified, default junction path will be used.

--policy
Specifies a policy to be applied for an AFM to cloud object storage fileset. If not specified, default policy is enabled.

--tmpdir
Specifies a directory pattern if a policy is not provided.

--quota-files
Specifies the number of file limit to be set as file quota on an AFM to cloud object storage fileset. If not specified, default file quota is in effect. Units are in numbers, for example, 104857600.

--quota-blocks
Specifies the data block limits to be set as block quota on an AFM to cloud object storage fileset. If not specified, default block quota is in effect. Units are in numbers.

--uid
Specifies a user ID to be set on an AFM to cloud object storage fileset. If not specified, default owner will be set, for example, root.

--gid
Specifies a group ID to be set on an AFM to cloud object storage fileset. If not specified, default group will be set, for example, root.

--perm
Specifies the access permission to be set on an AFM to cloud object storage fileset. This is in the octal format. Example 0770. If not specified, default permission will be set.

--mode
All AFM fileset modes support an AFM to cloud object storage fileset. These modes are independent-writer (IW), single-writer (SW), read-only (RO), and local-updates (LU). Default is the SW mode.

--chunk-size
Specifies the chunk size to control the number of upload parts on a cloud object storage. The unit is defined in number. Default is 16 MB chunk size.

--read-size
Specifies the read size to control the number of download data blocks into an AFM to cloud object storage fileset. The unit is defined in number. Default is 16 MB download size.

--acls
Enables the cache to synchronize ACL to the cloud object storage server. If not specified, AFM will not synchronize ACLs to cloud object storage.

Default is disabled ACL synchronization.

--report
This option is used to generate a colon (':') separated information of the AFM to cloud object storage enabled fileset. This option shows information such as policy, quota that is set on the fileset. This option generates a report of the specified bucket and serverName combination.

To change the configuration or a parameter of an AFM to cloud object storage fileset that is already created, you need to run separate commands such as **mmchfileset** and **mmchpolicy**.

Exit status

0
Successful completion.

nonzero
A failure occurred.

Security

You must have root authority to run the `mmafmcosconfig` command.

The node on which the command is issued must be able to run remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see the topic *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

Examples

1. To create an AFM to cloud object storage fileset by using the **ObjectFS** mode, issue the following command:

```
# mmafmcosconfig fs1 SW1 --endpoint http://<IP>:<port> --xattr --bucket vault1 --mode sw
--object-fs
```

2. To create an AFM to cloud object storage fileset by using the **ObjectOnly** mode, issue the following command:

```
# mmafmcosconfig fs1 SW2 --endpoint http://<IP>:<port> --xattr --bucket vault2 --mode sw
```

3. To list the fileset report in the colon (:) separate format, issue the following command:

```
# mmafmcosconfig fs1 SW2  --report
```

See also

- “mmafmconfig command” on page 45
- “mmafmcosaccess command” on page 48
- “mmafmcosctl command” on page 55
- “mmafmcoskeys command” on page 58
- “mmafmlocal command” on page 78
- “mmchattr command” on page 157
- “mmchconfig command” on page 170
- “mmchfileset command” on page 224
- “mmchfs command” on page 232
- “mmcrfileset command” on page 311
- “mmcrfs command” on page 318
- “mmlsconfig command” on page 495
- “mmlsfileset command” on page 501
- “mmlsfs command” on page 506
- “mmpsnap command” on page 614

For more information, see the AFM, AFM-based DR, and AFM to cloud object storage chapters in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Location

`/usr/lpp/mmfs/bin`

mmafmcosctl command

Controls downloads and uploads between a cloud object storage and an AFM to cloud object storage fileset.

Synopsis

```
mmafmcosctl Device FilesetName Path download {--object-list ObjectList | --all}
          [{--metadata | --data} --no-sub-dir --prefix NamePrefix --uid UID --gid GID --perm Permission]
```

or

```
mmafmcosctl Device FilesetName Path upload {--object-list ObjectList | --all}
```

or

```
mmafmcosctl Device FilesetName Path evict {--object-list ObjectList | --all} [--metadata]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

This command is used to control upload, download, and eviction on an AFM to cloud object storage fileset. For more information, see *Filesets* in the *IBM Spectrum Scale: Administration Guide*.

Parameters

Device
Specifies a device name of a file system.

FilesetName
Specifies a name of an existing AFM to cloud object storage fileset on which the operation is to be performed.

Path
Specifies a path under an AFM to cloud object storage fileset on which the operation needs to be performed.

download
Specifies the download operation that is performed on an AFM to cloud object storage fileset. This operation downloads the objects from a cloud object storage and presents them as files to the AFM to cloud object storage fileset.

upload
Specifies the upload operation that is performed on an AFM to cloud object storage fileset.

On a local-update (LU)-mode fileset that is enabled with the AFM to cloud object storage, you can upload data that was created or modified on the fileset locally (on the cache). You can upload this data to the cloud object storage server as an object. This is supported for the LU mode.

evict
Specifies the eviction operation of the file data (blocks) or file metadata (inode) on an AFM to cloud object storage fileset.

This operation removes file data and/or file metadata (inodes) only from the AFM to cloud object storage. This operation removes only data on AFM to cloud object storage. Objects on AFM to cloud object storage are not removed. This is a local operation.

This operation helps to save space (storage blocks and inodes) so that less priority data will be evicted from the local cache and makes space for priority data as per the quota. You can either evict all

files and/or metadata by using the `--all` option or evict selected data or metadata by using the `--object-list` option. The objectlist file is a line separated file that contains a full path of files on the fileset.

If require, you can download objects to the AFM to cloud object storage fileset again by using the `mmafmcosctl download` command.

Eviction will be triggered on the local cache and the objects on the cloud object storage will still have the data or metadata. This is a local operation.

--object-list
Specifies a line separated list of files that you want to download, upload, or evict. Use the option perform operation on selected files. Otherwise, you can use the `--all` option. This list file of objects might contain relative paths that support upload and download operations.

When you use the evict operation, a full file path name must be specified. You can either specify the `--object-list` or `--all` option.

--all
Specified with upload, download and evict parameters. This option performs the upload, download, or evict operation on all the data inside the given fileset. AFM generates internally a list of all the files and performs the operation on the list.

--metadata
Specified with download and evict operations. The option enables the AFM to cloud object storage to download or evict metadata.

When this option is specified with the download, only the objects metadata information is downloaded from the cloud object storage. This helps to populate all the directory tree structure on the cache from the bucket.

You can run the download operation along with the `--data` option to populate data blocks. This is the default option for downloading objects.

When this option is specified with the evict operation, both the data blocks and file inodes are evicted only from the AFM fileset. However, data blocks and file inode information will still be available in the bucket on cloud object storage server. This makes the space on the cache fileset for required data. This parameter is optional for the evict operation. If this option is not specified for eviction, default is to evict data blocks.

--data
Specified with the download operation. This option enables the AFM to cloud object storage to download data of the given list. The `--data` option downloads metadata implicitly.

--no-sub-dir
Specifies the AFM to cloud object storage to skip downloading the sub directories from a cloud object storage. You can skip this option to download the sub directories from a cloud object storage.

--prefix
Specifies a prefix for an object on a specified directory on an AFM to cloud object storage fileset.

--uid
Specifies a user ID to be set on an AFM to cloud object storage fileset. If not specified, the owner is set as default, for example, root.

--gid
Specifies a group ID to be set on an AFM to cloud object storage fileset. If not specified, the group is set as default, for example, root.

--perm
Specifies the access permission to be set on an AFM to cloud object storage fileset. This is in the octal format, for example, 0770. If not specified, the permission is set to default.

Exit status

0
Successful completion.

nonzero
A failure occurred.

Security

The `mmafmcosctl` command execution does not require a root user. This command enables non-root users to download, upload, or evict data from an AFM to cloud object storage fileset.

The node on which the command is issued must be able to run remote shell commands on any other node in the cluster. The node must run these remote shell commands without a password and must not produce any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

Examples

1. To download data of selected objects from a bucket to the cache fileset by using the object-list file, issue the following command:

```
# mmafmcosctl fs1 SW1 /gpfs/fs1/SW1 download --object-list /home/user01/listfile --data
```

2. To download all objects from a bucket to the cache, issue the following command:

```
# mmafmcosctl fs1 SW1 /gpfs/fs1/SW1 download --all --metadata --uid user01 --gid gid01
```

3. To evict files or objects data, issue the following command:

```
# mmafmcosctl fs1 afmtocos1 /gpfs/fs1/afmtocos1/ evict --object-list /root/evictfile
```

4. To evict files or objects metadata, issue the following command:

```
# mmafmcosctl fs1 afmtocos1 /gpfs/fs1/afmtocos1/ evict --object-list /root/evictfile --
metadata
```

5. To download files by using the **--prefix** parameter, issue the following command:

```
# mmafmcosctl fs1 iw1 /gpfs/fs1/iw1/ download --object-list /root/downloadfiles --data --
prefix thread_1
```

A sample output is as follows:

```
     Queued    (Total)      Failed                TotalData
                                          (approx in Bytes)
          0         (0)          0                      0
         20         (0)          0                 204800
Object Downloads successfully queued at the gateway.
```

A prefix is a directory inside the `/gpfs/fs1/iw1` fileset, and the object-list contains a path for all files under the prefix.

See also

- “mmafmcosaccess command” on page 48
- “mmafmcosconfig command” on page 50
- “mmafmcoskeys command” on page 58

Location

`/usr/lpp/mmfs/bin`

mmafmcoskeys command

Manages an access key and a secret key to access a bucket on a cloud object storage.

Synopsis

```
mmafmcoskeys bucket[:{[Region@]Server | ExportMap}]
             {set {akey skey | --keyfile filePath} | get | delete}
```

or

```
mmafmcoskeys all get --report
```

Availability

Available on all IBM Spectrum Scale editions.

Description

This command manages an access key and a secret key to access a bucket on a cloud object storage for data synchronization. This command can set, get, delete, and report the access key and the secret key credentials. The keys that are stored for a bucket by AFM have a unique identity across the cluster. If a bucket name is common across multiple cloud object storage servers, the keys can be set along with the server name.

The keys can either be specified by using the command line or can be provided as an input key file in the format of colon delimiter in each line for `akey` and `skey`. A report can be displayed to list all access keys or secret keys that are stored for a bucket. This report has a list of all keys across the cluster.

Parameters

bucket
Identifies a unique bucket on a cloud object storage. AFM will use this bucket as a target for a fileset and it will synchronize data between the bucket and the fileset by using the keys that are provided.

Note:

- Ensure that the name of bucket is as per the support of the cloud object storage. You need to check the cloud object storage for bucket names guidelines.
- A duplicate combination of a bucket name and a server name is not allowed.

Server
If the same bucket exists on multiple cloud object storage, the bucket credentials can be stored by providing the `bucket:serverName` format. The combination of a bucket name and a server name will uniquely identify the keys of the bucket across the cluster. AFM shows a reject message if you try to store duplicate bucket name.

Region
Specifies a geographical region of a bucket. Provide the region if the bucket is hosted on a server in a non-default region. You need not to set a region for a default bucket region.

If region is not specified during the creation of a cloud object storage fileset, AFM tries to access a bucket on a server in the default region.

You must set a region if a bucket is hosted on a server in a non-default region.

set
Stores the bucket credentials, which are an access key and a secrete key, with AFM. AFM stores this credential for communication with a cloud object storage. Wrong credentials are validated only at the time of communication.

Access and secret keys can be provided to AFM either by using command line where you can use a space separated word or by using a key file, which has separate lines for `akey:`*`AccessKey`* and `skey:`*`SecretKey`*.

This command can be used to modify the stored credentials that are used for next communication with a cloud object storage. It is recommended to modify credentials when the communication between AFM and a cloud object storage is stopped.

You must store credentials of a bucket before you create an AFM fileset that has a cloud object storage as a target.

get
Retrieves the stored credentials that were set for the bucket and server combination. AFM shows an error if the bucket and server name combination does not exist.

delete
Deletes the credentials of the bucket and server name combination that is stored with AFM. The `delete` operation removes entries of access keys and secret keys of the specified bucket. Before you delete the stored credentials of a bucket, you need to ensure that the bucket is not mentioned as a target on any AFM fileset.

--akey *AccessKey*
Specifies an access key that belongs to a cloud object storage that is used to access the bucket.

--skey *SecretKey*
Specifies a secret key that belongs to a cloud object storage that is used to access the bucket.

--keyfile
Specifies a key file that contains an access key and a secret key. Instead of providing the access key and the secret key on the command line, a key file can be used. The key file must contain two lines for `akey` and `skey` separated by a colon. An example of the format of a key file */root/keyfile1* is as follows:

```
akey:AccessKey
skey:SecretKey
```

--report
This option generates a report of all buckets and server name combinations that are stored with AFM.

Exit status

0
Successful completion.

nonzero
A failure occurred.

Security

You must have root authority to run the `mmafmcoskeys` command.

The node on which this command is issued must be able to run remote shell commands on any other node in the cluster. The node must run these remote shell commands without a password and must not produce any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

Examples

1. To set an access key and a secret key of a bucket, issue the following the command:

```
# mmafmcoskeys bucket1 set AccessKey SecretKey
```

2. To set an access key and a secret key of a bucket by using a key file, issue the following command:

An example of a key file in the */root/key* file is as follows:

```
akey:AccessKey
skey:SecretKey
```

Note: The key file must not have any other characters such as space and tab.

```
# mmafmcoskeys bucket1 set --keyfile /root/keyfile
```

3. To set an access key and a secret key of a bucket by using the `bucket:serverName` combination, issue the following command:

```
# mmafmcoskeys bucket1:serverName1 set AccessKey SecretKey
```

4. To get an access key and a secret key of a bucket, issue the following command:

```
# mmafmcoskeys bucket1 get
```

5. To get an access key and a secret key of a bucket by using the `bucket:serverName` combination, issue the following command:

```
# mmafmcoskeys bucket1:serverName1 get
```

6. To delete an access key and a secret key of a bucket, issue the following command:

```
# mmafmcoskeys bucket1 delete
```

7. To delete an access key and a secret key of a bucket by using the `bucket:serverName` combination, issue the following command:

```
# mmafmcoskeys bucket1:serverName1 delete
```

8. To set an access key and a secret key of a bucket by using a bucket and a server name combination, issue the following command:

```
# mmafmcoskeys bucket2:us-west-2@ServerName2 set AccessKey SecretKey
```

9. To get an access key and a secret key of a bucket, issue the following command:

```
# mmafmcoskeys bucket2 get
```

10. To get and an access key and a secret key of a non-default region bucket, issue the following command:

```
# mmafmcoskeys bucket2:us-west-2@ServerName2 get
```

11. To generate a report of all buckets and their credentials that are stored, issue the following command:

```
# mmafmcoskeys all get --report
```

See also

- “mmafmcosaccess command” on page 48
- “mmafmcosconfig command” on page 50
- “mmafmcosctl command” on page 55

Location

`/usr/lpp/mmfs/bin`

mmafmctl command

This command is for various operations and reporting information on all filesets. It is recommended to read the *IBM Spectrum Scale: Administration Guide* AFM and AFM Disaster Recovery chapters along with this manual for detailed description of the functions.

Synopsis

To use the AFM DR functions correctly, use all commands enlisted in this chapter in accordance with the steps described in the AFM-based DR chapter in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

AFM read only mode is referred as RO, single writer mode is referred as SW, independent writer mode is referred as IW, and local update mode is referred as LU in this manual.

```
mmafmctl Device {resync | expire | unexpire | stop | start} -j FilesetName
```

or

```
mmafmctl Device {getstate | resumeRequeued} -j FilesetName
```

or

```
mmafmctl Device flushPending [-j FilesetName [--list-file ListFile]]
         [-s LocalWorkDirectory]
```

or

```
mmafmctl Device failover -j FilesetName
         --new-target NewAfmTarget [--target-only] [-s LocalWorkDirectory]
```

or

```
mmafmctl Device prefetch -j FilesetName [-s LocalWorkDirectory]
         [--retry-failed-file-list | --enable-failed-file-list]
         [{--directory LocalDirectoryPath | --dir-list-file DirListFile [--policy]} [--
nosubdirs]]
         [{--list-file ListFile | --home-list-file HomeListFile} [--policy]]
         [--home-inode-file PolicyListFile]
         [--home-fs-path HomeFileSystemPath]
         [--metadata-only] [--gateway Node]
         [--readdir-only] [--force] [--prefetch-threads nThreads]
```

or

```
mmafmctl Device evict -j FilesetName
         [--safe-limit SafeLimit] [--order {LRU | SIZE}]
         [--log-file LogFile] [--filter Attribute=Value ...]
         [--list-file Listfile] [--file FilePath]
```

or

```
mmafmctl Device failback -j FilesetName {{--start --failover-time Time} | --stop}
         [-s LocalWorkDirectory]
```

or

```
mmafmctl Device failoverToSecondary -j FilesetName [--norestore | --restore]
```

or

```
mmafmctl Device convertToPrimary -j FilesetName
         [--afmtarget Target {--inband | --secondary-snapname SnapshotName}]
         [--check-metadata | --nocheck-metadata ] [--rpo RPO] [-s LocalWorkDirectory]
```

or

```
mmafmctl Device convertToSecondary -j FilesetName --primaryid PrimaryId [--force]
```

or

```
mmafmctl Device changeSecondary -j FilesetName --new-target NewAfmTarget
         [--target-only | --inband] [-s LocalWorkDirectory]
```

or

```
mmafmctl Device replacePrimary -j FilesetName
```

or

```
mmafmctl Device failbackToPrimary -j FilesetName {--start | --stop} [--force]
```

or

```
mmafmctl Device {applyUpdates | getPrimaryId} -j FilesetName
```

Availability

Available on all IBM Spectrum Scale editions. Available on AIX and Linux.

Description

The usage of options of this command for different operations on both AFM (RO/SW/IW/LU) filesets and AFM primary/secondary filesets are explained with examples.

File system should be mounted on all gateway nodes for `mmafmctl` functions to work.

Parameters

Device
Specifies the device name of the file system.

-j *FilesetName*
Specifies the fileset name.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

-s *LocalWorkDirectory*
Specifies the temporary working directory.

1. This section describes:

```
mmafmctl Device {resync | expire | unexpire | stop|start} -j FilesetName
```

resync
This option is available only for SW cache. In case of inadvertent changes made at home of an SW fileset, such as delete of a file or change of data in a file etc., the administrator can correct the home by sending all contents from cache to home using this option. The limitation of this option that renamed files at home may not be fixed by `resync`. Using `resync` requires the cache to be either in `NeedsResync` or `Active` state.

expire | unexpire
This option is available only for RO cache to manually expire or unexpire. When an RO cache is disconnected, the cached contents are still accessible for the user. However, the administrator can define a timeout from home beyond which access to the cached contents becomes stale. Such an event would occur automatically after disconnection (when cached contents are no longer accessible) and is called *expiration*; the cache is said to be expired. This option is used to manually force the cache state to 'Expired'. To expire a fileset manually, the **afmExpirationTimeout** must be set on the fileset.

When the home comes back or reconnects, the cache contents become automatically accessible again and the cache is said to un-expire. The `unexpire` option is used to force cache to come out of the 'Expired' state.

The manual expiration and un-expiration can be forced on a cache even when the home is in a connected state. If a cache is expired manually, the same cache must be unexpired manually.

stop
Run on an AFM or AFM DR fileset to stop replication. You can use this command during maintenance or downtime, when the I/O activity on the filesets is stopped, or is minimal. After the fileset moves to a 'Stopped' state, changes or modifications to the fileset are not sent to the gateway node for queuing.

start
Run on a 'Stopped' AFM or AFM DR fileset to start sending updates to the gateway node and resume replication on the fileset.

2. This section describes:

```
mmafmctl Device {getstate | resumeRequeued} -j FilesetName
```

getstate
This option is applicable for all AFM (RO/SW/IW/LU) and AFM primary filesets. It displays the status of the fileset in the following fields:

Fileset Name
The name of the fileset.

Fileset Target
The host server and the exported path on it.

Gateway Node
Primary gateway of the fileset. This gateway node is handling requests for this fileset.

Queue Length
Current length of the queue on the primary gateway.

Queue numExec
Number of operations played at home since the fileset is last `Active`.

Cache State

- Cache states applicable for all AFM RO/SW/IW/LU filesets:

  `Active`, `Inactive`, `Dirty`, `Disconnected`, `Stopped`, `Unmounted`
- Cache states applicable for RO filesets:

  `Expired`
- Cache states applicable for SW and IW filesets:

  `Recovery`, `FlushOnly`, `QueueOnly`, `Dropped`, `NeedsResync`, `FailoverInProgress`
- Cache states applicable for IW filesets:

  `FailbackInProgress`, `FailbackCompleted`, `NeedsFailback`
- Cache states applicable for AFM primary filesets:

  `PrimInitInProg`, `PrimInitFail`, `Active`, `Inactive`, `Dirty`, `Disconnected`, `Unmounted`, `FailbackInProg`, `Recovery`, `FlushOnly`, `QueueOnly`, `Dropped`, `Stopped`, `NeedsResync`

For more information about all cache states, see the AFM and AFM-based DR chapters in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

resumeRequeued
This option is applicable for SW/IW and primary filesets. If there are operations in the queue that were re-queued due to errors at home, the Administrator should correct those errors and can run this option to retry the re-queued operations.

3. This section describes:

```
mmafmctl Device flushPending [-j FilesetName [--list-file ListFile]]
            [-s LocalWorkDirectory]
```

flushPending
Flushes all point-in-time pending messages in the normal queue on the fileset to home. Requeued messages and messages in the priority queue for the fileset are not flushed by this command.

When `--list-file` *ListFile* is specified, the messages pending on the files listed in the list file are flushed to home. *ListFile* contains a list of files that you want to flush, one file per line. All files must have absolute path names, specified from the fileset linked path. If the list of files has filenames with special characters, use a policy to generate the listfile. Edit to remove all entries other than the filenames. FlushPending is applicable for SW/IW and primary filesets.

4. This section describes:

```
mmafmctl Device failover -j FilesetName
            --new-target NewAfmTarget [--target-only] [-s LocalWorkDirectory]
```

This option is applicable only for SW/IW filesets. This option pushes all the data from cache to home. It should be used only in case home is completely lost due to a disaster and a new home is being set up. Failover often takes a long time to complete; status can be checked by using the `afmManualResyncComplete` callback or via **`mmafmctl getstate`** command.

--new-target *NewAfmTarget*
Specifies a new home server and path, replacing the home server and path originally set by the `afmTarget` parameter of the `mmcrfileset` command. Specified in either of the following formats:

```
nfs://{Host|Map}/Target_Path
```

or

```
gpfs://[Map]/Target_Path
```

where:

nfs:// or gpfs://
Specifies the transport protocol.

Host*|*Map

Host
Specifies the server domain name system (DNS) name or IP address.

Map
Specifies the export map name. Information about Mapping is contained in the AFM Overview > Parallel data transfers section.

See the following examples:

1. An example of using the nfs:// protocol with a map name:

```
mmcrfileset fs3 singleWriter2 -p
        afmtarget=nfs://<map1>/gpfs/fs1/target1 -p afmmode=sw --inode-space new
```

2. An example of using the nfs:// protocol with a host name:

```
mmcrfileset fs3 singleWriter2 -p
        afmtarget=nfs://<hostname>/gpfs/fs1/target1 -p afmmode=sw --inode-space new
```

3. An example of using the gpfs:// protocol without a map name:

```
mmcrfileset fs3 singleWriter1 -p
         afmtarget=gpfs:///gpfs/thefs1/target1 -p afmmode=sw --inode-space new
```

Note: If you are not specifying the map name, a '/' is still needed to indicate the path.

4. An example of using the gpfs:// protocol with a map name:

```
mmcrfileset fs3 singleWriter1 -p
        afmtarget=gpfs://<map>/gpfs/thefs1/target1 -p afmmode=sw --inode-space new
```

It is possible to change the protocol along with the target using failover. For example, a cache using an NFS target `bear110:/gpfs/gpfsA/home` can be switched to a GPFS target whose remote file system is mounted at `/gpfs/fs1`, and vice-versa, as follows:

```
mmafmctl fs0 failover -j afm-mc1 --new-target gpfs:///gpfs/fs1
mmafmctl fs0 failover -j afm-mc1 --new-target nfs://bear110/gpfs/gpfsA/home
```

In the first command, `///` is needed because *Host* is not provided.

--target-only
This option is used to change the mount path or IP address in the target path. The new NFS server must be in the same home cluster and must have the same architecture as the existing NFS server in the target path. This option must not be used to change the target location or protocol. You must ensure that the new NFS server exports the same target path that has the same FSID.

5. This section describes:

```
mmafmctl Device prefetch -j FilesetName [-s LocalWorkDirectory]
             [--retry-failed-file-list|--enable-failed-file-list]
             [ {--directory LocalDirectoryPath | --dir-list-file DirListfile [--policy]} [--
nosubdirs]]
             [{--list-file ListFile | --home-list-file HomeListFile} [--policy]]
             [--home-inode-file PolicyListFile]
             [--home-fs-path HomeFilesystemPath]
             [--metadata-only] [--gateway Node]
             [--readdir-only] [--force] [--prefetch-threads nThreads]
```

This option is used for pre-fetching file contents from home before the application requests for the contents. This reduces the network delay when the application performs data transfers on file and data that is not in cache. You can also use this option to move files over the WAN when the WAN usage is low. These files might be the files that are accessed during high WAN usage. Thus, you can use this option for better WAN management.

Prefetch is an asynchronous process and you can use the fileset when prefetch is in progress. You can monitor Prefetch using the **afmPrepopEnd** event. AFM can prefetch the data using the **mmafmctl prefetch** command (which specifies a list of files to prefetch). Prefetch always pulls the complete file contents from home and AFM automatically sets a file as cached when it is completely prefetched.

You can use the prefetch option to -

- populate metadata
- populate data
- view prefetch statistics

Prefetch completion can be monitored using the `afmPrepopEnd` event.

--retry-failed-file-list
Allows retrying prefetch of files that failed in the last prefetch operation. The list of files to retry is obtained from `.afm/.prefetchedfailed.list` under the fileset.

Note: To use this option, you must enable generating a list of failed files. Add **`--enable-failed-file-list`** to the command first.

--metadata-only
Prefetches only the metadata and not the actual data. This is useful in migration scenarios. This option requires the list of files whose metadata you want. Hence it must be combined with a list file option.

--enable-failed-file-list
Turns on generating a list of files which failed during prefetch operation at the gateway node. The list of files is saved as `.afm/.prefetchedfailed.list` under the fileset. Failures that occur during processing are not logged in `.afm/.prefetchedfailed.list`. If you observe any errors during processing (before queuing), you might need to correct the errors and rerun prefetch.

Files listed in `.afm/.prefetchedfailed.list` are used when prefetch is retried.

--policy
Specifies that the `list-file` or `home-list-file` is generated using a GPFS Policy by which sequences like '\' or '\n' are escaped as '\\' and '\\n'. If this option is specified, input file list is treated as already escaped. The sequences are unescaped first before queuing for prefetch operation.

Note: This option can be used only if you are specifying `list-file` or `home-list-file`.

--directory *LocalDirectoryPath*
Specifies path to the local directory from which you want to prefetch files. A list of all files in this directory and all its sub-directories is generated, and queued for prefetch.

--dir-list-file *DirListfile*
Specifies path to the file which contains unique entry of directories under the AFM fileset which needs to be prefetched. This option enables to prefetch individual directories under an AFM fileset. AFM generate a list of all files and sub-directories inside and queued for prefetch. The input file could also be a policy generated file for which user needs to specify `--policy`.

You can either specify *--directory* or *--dir-list-file* option with **`mmafmctl prefetch`**.

The `--policy` option can be used only with `--dir-list-file` and not with *--directory*.

For example,

```
mmafmctl fs1 prefetch -j fileset1 --dir-list-file /tmp/file1 --policy --nosubdirs
```

--nosubdirs
This option restricts the recursive behavior of `--dir-list-file` and *--directory* and prefetch only until the given level of directory. This option will not prefetch the sub-directories under the given directory. This is optional parameter.

This option can only be used with `--dir-list-file` and *--directory*.

For example,

```
#mmafmctl fs1 prefetch -j fileset1 --directory  /gpfs/fs1/fileset1/dir1 --nosubdirs

#mmafmctl fs1 prefetch -j fileset1 --dir-list-file  /tmp/file1 --policy --nosubdirs
```

--list-file *ListFile*
The specified file is a file containing a list of files, and needs to be prefetched, one file per line. All files must have fully qualified path names.

If the list of files to be prefetched have filenames with special characters then a policy must be used to generate the listfile. Remove entries from the file other than the filenames.

An indicative list of files:

- files with fully qualified names from cache
- files with fully qualified names from home
- list of files from home generated by using policy. Do not edit.

--home-list-file ***HomeListFile***
Contains a list of files from the home cluster that needs to be prefetched, one file per line. All files must have fully qualified path names. If the list of files has filenames with special characters, use a policy to generate the `listfile`. Edit to remove all entries other than the `filenames`.

This command is deprecated. Use **`-list-file`** instead.

--home-inode-file ***PolicyListFile***
Contains a list of files from the home cluster that needs to be prefetched in the cache. Do not edit the file. The file is generated by using policy.

This command is deprecated. Use **`-list-file`** instead.

--home-fs-path ***HomeFileSystemPath***

Specifies the full path to the fileset at the home cluster and can be used along with *ListFile*.

You must use this option, when in the NSD protocol the mount point on the gateway nodes of the **`afmTarget`** filesets does not match the mount point on the Home cluster.

For example, **`mmafmctl gpfs1 prefetch -j cache1 –list-file /tmp/list.allfiles --home-fs-path /gpfs/remotefs1`**

In this example, the file system is mounted on the :

- home cluster at `/gpfs/homefs1`
- gateway nodes at `/gpfs/remotefs1`

--readdir-only
This option overrides the dirty flag that is set when the data is modified at the local LU cache. In the LU mode, the dirty flag does not allow the `readdir` operation at the home and refreshes the directory file entries from the home.

This option performs `readdir` one last time on the directory at the home after the data migration to the cache and the application is started on the cache data. When the application is moved to the cache, the application creates or modifies data at the cache. The data that is modified or created makes the directory dirty. Therefore, AFM never inquires and brings any latest changes from the home.

During the migration process to avoid any further `readdir` AFM file, directory refresh intervals can be disabled. When this option is set, no further `readdir` operations are allowed.

The **`afmReadDirOnce`** parameter must be set on the fileset and refresh intervals must be disabled.

For example,

```
# mmafmctl <fs> prefetch -j <fileset> --directory /<fileset_path>/<directory> --readdir-only
```

--force
Enables forcefully fetching data from the home during the migration process. This option overrides any set restrictions and helps to fetch the data forcefully to the cache. This option must be used only to forcefully fetch the data that was created after the migration process completion.

For example,

```
# mmafmctl <fs> prefetch -j <fileset> --list-file <listfile_path> --force
```

--gateway ***Node***
Allows selecting the gateway node that can be used to run the prefetch operation on a fileset, which is idle or less used. This parameter helps to distribute the prefetch work on different gateway nodes and overrides the default gateway node, which is assigned to the fileset. This parameter also helps to run different prefetch operations on different gateway nodes, which might belong to the same fileset or a different fileset.

For example,

```
# mmafmctl <fs> prefetch -j <fileset> --list-file <listfile_path> --gateway Node2
```

To check the prefetch statistics of this command on gateway Node2, issue the following command:

```
# mmafmctl <fs> prefetch -j <fileset> --gateway Node2
```

--prefetch-threads *nThreads*
Specifies the number of threads to be used for the prefetch operation. Valid values are 1 - 255. Default value is 4.

For example,

```
# mmafmctl <fs> prefetch -j <fileset> --list-file <listfile_path> --prefetch-threads 6
```

If you run `prefetch` without providing any options, it displays statistics of the last `prefetch` command that is run on the fileset.

If you run the prefetch command with data or metadata options, statistics like queued files, total files, failed files, total data (in Bytes) is displayed as in the following example of command and system output:

#mmafmctl <FileSystem> prefetch -j <fileset> --enable-failed-file-list --list-file /tmp/file-list

```
mmafmctl: Performing prefetching of fileset: <fileset>
Queued (Total) Failed TotalData (approx in Bytes)
0      (56324) 0      0
5      (56324) 2      1353559
56322  (56324) 2      14119335
```

6. This section describes:

```
mmafmctl Device evict -j FilesetName
            [--safe-limit SafeLimit] [--order {LRU | SIZE}]
            [--log-file LogFile] [--filter Attribute=Value ...]
            [--list-file ListFile] [--file FilePath]
```

This option is applicable for RO/SW/IW/LU filesets. When cache space exceeds the allocated quota, data blocks from non-dirty are automatically de-allocated with the eviction process. This option can be used for a file that is specifically to be de-allocated based on some criteria. All options can be combined with each other.

--safe-limit *SafeLimit*
This is a compulsory parameter for the manual evict option, for order and filter attributes. Specifies target quota limit (which is used as the low water mark) for eviction in bytes; must be less than the soft limit. This parameter can be used alone or can be combined with one of the following parameters (order or filter attributes). Specify the parameter in bytes.

--order LRU | SIZE
Specifies the order in which files are to be chosen for eviction:

LRU
Least recently used files are to be evicted first.

SIZE
Larger-sized files are to be evicted first.

--log-file *LogFile*
Specifies the file where the eviction log is to be stored. The default is that no logs are generated.

--filter *Attribute=Value*
Specifies attributes that enable you to control how data is evicted from the cache. Valid attributes are:

FILENAME=*FileName*
Specifies the name of a file to be evicted from the cache. This uses an SQL-type search query. If the same file name exists in more than one directory, it will evict all the files with that name. The complete path to the file should not be given here.

MINFILESIZE=*Size*
Sets the minimum size of a file to evict from the cache. This value is compared to the number of blocks allocated to a file (KB_ALLOCATED), which may differ slightly from the file size.

MAXFILESIZE=*Size*
Sets the maximum size of a file to evict from the cache. This value is compared to the number of blocks allocated to a file (KB_ALLOCATED), which may differ slightly from the file size.

--list-file *ListFile*
Contains a list of files that you want to evict, one file per line. All files must have fully qualified path names. File system quotas need not be specified. If the list of files has file names with special characters, use a policy to generate the `listfile`. Edit to remove all entries other than the file names.

--file *FilePath*
The fully qualified name of the file that needs to be evicted. File system quotas need not be specified.

Possible combinations of *safelimit*, *order*, and *filter* are:

only Safe limit
Safe limit + LRU
Safe limit + SIZE
Safe limit + FILENAME
Safe limit + MINFILESIZE
Safe limit + MAXFILESIZE
Safe limit + LRU + FILENAME
Safe limit + LRU + MINFILESIZE
Safe limit + LRU + MAXFILESIZE
Safe limit + SIZE + FILENAME
Safe limit + SIZE + MINFILESIZE
Safe limit + SIZE + MAXFILESIZE

7. This section describes:

```
mmafmctl Device failback -j FilesetName {{--start --failover-time Time} | --stop}
```

```
        [-s LocalWorkDirectory]
```

`failback` is applicable only for IW filesets.

failback --start --failover-time *Time*
Specifies the point in time at the home cluster, from which the independent-writer cache taking over as writer should sync up. The failover *Time* can be specified in date command format with time zones. It will use the cluster's time-zone and year by default.

failback `--stop`
An option to be run after the failback process is complete and the fileset moves to `FailbackCompleted` state. This option will move the fileset to `Active` state.

8. This section describes:

```
mmafmctl Device failoverToSecondary -j FilesetName [--norestore |--restore ]
```

This is to be run on a secondary fileset.

When primary experiences a disaster, all applications will need to be moved to the secondary to ensure business continuity. The secondary must be first converted to an acting primary using this option.

There is a choice of restoring the latest snapshot data on the secondary during the failover process or leave the data as is using the `--norestore` option. Once this is complete, the secondary becomes ready to host applications.

--norestore
Specifies that restoring from the latest RPO snapshot is not required. This is the default setting.

--restore
Specifies that data must be restored from the latest RPO snapshot.

9. This section describes:

```
mmafmctl Device convertToPrimary -j FilesetName
         [ --afmtarget Target { --inband | --secondary-snapname SnapshotName }]
         [ --check-metadata | --nocheck-metadata ] [--rpo RPO] [-s LocalWorkDirectory]
```

This is to be run on a GPFS fileset or SW/IW fileset which is intended to be converted to primary.

--afmtarget*Target*
Specifies the secondary that needs to be configured for this primary. Need not be used for AFM filesets as target would already have been defined.

--inband
Used for inband trucking. *Inband trucking* copies data from the primary site to an empty secondary site during conversion of GPFS filesets to AFM DR primary filesets. If you have already copied data to the secondary site, AFM checks **mtime** of files at the primary and secondary site. Here, granularity of **mtime** is in microseconds. If **mtime** values of both files match, data is not copied again and existing data on the secondary site is used. If **mtime** values of both files do not match, existing data on the secondary site is discarded and data from the primary site is written to the secondary site.

--check-metadata
This is the default option. Checks if the disallowed types (like immutable/append-only files) are present in the GPFS fileset on the primary site before the conversion. Conversion with this option fails if such files exist.

For SW/IW filesets, presence of orphans and incomplete directories are also checked. SW/IW filesets should have established contact with at least once home for this option to succeed.

--nocheck-metadata
Used if one needs to proceed with conversion without checking for appendonly/immutable files.

--secondary-snapname *SnapshotName*
Used while establishing a new primary for an existing secondary or acting primary during failback.

--rpo *RPO*
Specifies the RPO interval in minutes for this primary fileset. Disabled by default.

10. This section describes:

```
mmafmctl Device convertToSecondary -j FilesetName --primaryid  PrimaryId [ --force ]
```

This is to be run on a GPFS fileset on the secondary site. This converts a GPFS independent fileset to a secondary and sets the primary ID.

--primaryid *PrimaryId*
Specifies the unique identifier of the AFM-DR primary fileset which needs to be set at AFM-DR Secondary fileset to initiate a relationship. You can obtain this fileset identifier by running the **mmlsfileset** command using the `--afm` and `-L` options.

For example,

```
#mmlsfileset <FileSystem> <AFM DR Fileset> -L --afm |grep 'Primary Id'
```

--force
If `convertToSecondary` failed or got interrupted, it will not create afmctl file at the secondary. The command should be rerun with the `--force` option.

11. This section describes:

```
mmafmctl Device changeSecondary -j FilesetName
```

```
--new-target NewAfmTarget [ --target-only |--inband ]
         [-s LocalWorkDirectory]
```

This is to be run on a primary fileset only.

A disaster at the secondary can take place due to which secondary is not available.

Run this command on the primary when a secondary fails and this primary needs to be connected with a new secondary. On the new secondary site a new GPFS independent fileset has to be created. Data on the primary can be copied to the new GPFS fileset that was created with this command using other means such as `ftp`, `scp`. Alternatively it can be decided that data will be trucked using the relationship.

--new-target *NewAfmTarget*
Used to mention the new secondary.

--inband
Used for inband trucking. Copies data to a new empty secondary. If you have already copied data to the secondary site, **mtime** of files at the primary and secondary site is checked. Here, granularity of **mtime** is in microseconds. If **mtime** values of both files match, data is not copied again and existing data on the secondary site is used. If **mtime** values of both files do not match, existing data on the secondary site is discarded and data from the primary site is written to the secondary site.

--target-only

Used when you want to change the IP address or NFS server name for the same target path. The new NFS server must be in the same home cluster and must be of the same architecture(power or x86) as the existing NFS server in the target path. This option can be used to move from NFS to a mapping target.

12. This section describes:

```
mmafmctl Device replacePrimary -j FilesetName
```

This is used on an acting primary only. This creates a latest snapshot of the acting primary. This command deletes any old RPO snapshots on the acting primary and creates a new initial RPO snapshot `psnap0`.

This RPO snapshot is used in the setup of the new primary.

13. This section describes:

```
mmafmctl Device failbackToPrimary -j FilesetName {--start | --stop} [--force]
```

This is to be run on an old primary that came back after the disaster, or on a new primary that is to be configured after an old primary went down with a disaster. The new primary should have been converted from GPFS to primary using convertToPrimary option.

--start
Restores the primary to the contents from the last RPO on the primary before the disaster. This option will put the primary in read-only mode to avoid accidental corruption until the failback process is completed. In case of new primary that is setup using `convertToPrimary`, the `failback --start` does no change.

--stop
Used to complete the Failback process. This will put the fileset in read-write mode. The primary is now ready for starting applications.

--force
Used if `--stop` or `--start` does not complete successfully due to any errors, and not allow `failbackToPrimary` to stop or start again.

14. This section describes:

```
mmafmctl Device {applyUpdates |getPrimaryId } -j FilesetName
```

Both options are intended for the primary fileset.

applyUpdates
Run this on the primary after running the `failback --start` command. All the differences can be brought over in one go or through multiple iterations. For minimizing application downtime, this command can be run multiple times to bring the primary's contents in sync with the acting primary. When the contents are as close as possible or minimal, applications should take a downtime and then this command should be run one last time.

It is possible that `applyUpdates` fails with an error during instances when the acting primary is overloaded. In such cases, the command must be run again.

getPrimaryID
Used to get primary ID of a primary fileset.

Exit status

0
Successful completion.

nonzero
A failure occurred.

Security

You must have root authority to run the `mmafmctl` command.

The node on which the command is issued must be able to run remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see the topic *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

Examples

1. **Running resync on SW:**

```
# mmafmctl fs1 resync -j sw1
mmafmctl: Performing resync of fileset: sw1

# mmafmctl fs1 getstate -j sw1
Fileset Name Fileset Target                       Cache State Gateway Node Queue Length Queue numExec
------------ --------------                       ----------- ------------ ------------ -------------
sw1          nfs://c26c3apv2/gpfs/homefs1/newdir1 Dirty       c26c2apv1    4067         10844
```

2. **Expiring a RO fileset:**

```
# mmafmctl fs1 expire -j ro1

# mmafmctl fs1 getstate -j ro1
Fileset Name Fileset Target             Cache State Gateway Node Queue Length Queue numExec
------------ --------------             ----------- ------------ ------------ -------------
ro1          gpfs:///gpfs/remotefs1/dir1 Expired     c26c4apv1    0            4
```

3. **Unexpiring a RO fileset:**

```
# mmafmctl fs1 unexpire -j ro1

# mmafmctl fs1 getstate -j ro1
Fileset Name Fileset Target             Cache State Gateway Node Queue Length Queue numExec
------------ --------------             ----------- ------------ ------------ -------------
ro1          gpfs:///gpfs/remotefs1/dir1 Active      c26c4apv1    0            4
```

4. **Run flushPending on SW fileset:**

```
// Populate the fileset with data

# mmafmctl fs1 getstate -j sw1
```

```
Fileset Name Fileset Target             Cache State Gateway Node Queue Length Queue numExec
------------ --------------             ----------- ------------ ------------ -------------
sw1          gpfs:///gpfs/remotefs1/dir1 Dirty       c26c2apv1    5671         293

Get the list of files newly created using policy:
RULE EXTERNAL LIST 'L' RULE 'List' LIST 'L' WHERE PATH_NAME LIKE '%'

# mmapplypolicy /gpfs/fs1/sw1/migrateDir.popFSDir.22655 -P p1 -f p1.res -L 1 -N mount -I defer

Policy created this file, this should be hand-edited to retain only the names:
11012030 65537 0 -- /gpfs/fs1/sw1/migrateDir.popFSDir.22655/file_with_posix_acl1
11012032 65537 0 -- /gpfs/fs1/sw1/migrateDir.popFSDir.22655/populateFS.log
11012033 65537 0 --
/gpfs/fs1/sw1/migrateDir.popFSDir.22655/sparse_file_0_with_0_levels_indirection

# cat p1.res.list | awk '{print $5}' > /lfile

# mmafmctl fs1 flushPending -j sw1 --list-file=/lfile
```

5. **Failover of SW to a new home:**

```
# mmafmctl fs1 getstate -j sw1
Fileset Name Fileset Target             Cache State Gateway Node Queue Length Queue numExec
------------ --------------             ----------- ------------ ------------ -------------
sw1          gpfs:///gpfs/remotefs1/dir1 Dirty       c26c2apv1    785          5179
```

At home -

```
# mmcrfileset homefs1 newdir1 --inode-space=new
Fileset newdir1 created with id 219 root inode 52953091.

# mmlinkfileset homefs1 newdir1 -J /gpfs/homefs1/newdir1
Fileset newdir1 linked at /gpfs/homefs1/newdir1

# mmafmconfig enable /gpfs/homefs1/newdir1
```

At cache -

```
# mmafmctl fs1 failover -j sw1 --new-target=c26c3apv1:/gpfs/homefs1/newdir1
mmafmctl: Performing failover to nfs://c26c3apv1/gpfs/homefs1/newdir1
Fileset sw1 changed.
mmafmctl: Failover in progress. This may take while...
 Check fileset state or register for callback to know the completion status.


Callback registered, logged into mmfs.log:
Thu May 21 03:06:18.303 2015: [I] Calling User Exit Script callback7: event
afmManualResyncComplete, Async command recovery.sh


# mmafmctl fs1 getstate -j sw1
Fileset Name Fileset Target                       Cache State Gateway Node Queue Length Queue numExec
------------ --------------                       ----------- ------------ ------------ -------------
sw1          nfs://c26c3apv1/gpfs/homefs1/newdir1 Active      c26c2apv1    0            5250
```

6. **Changing target of SW fileset:**

```
Changing to another NFS server in the same home cluster using --target-only option:

# mmafmctl fs1 failover -j sw1 --new-target=c26c3apv2:/gpfs/homefs1/newdir1 --target-only
mmafmctl: Performing failover to nfs://c26c3apv2/gpfs/homefs1/newdir1
Fileset sw1 changed.


# mmafmctl fs1 getstate -j sw1
Fileset Name Fileset Target                       Cache State Gateway Node Queue Length Queue numExec
------------ --------------                       ----------- ------------ ------------ -------------
sw1          nfs://c26c3apv2/gpfs/homefs1/newdir1 Active      c26c2apv1    0            5005
```

7. **Metadata population using prefetch:**

```
# mmafmctl fs1 getstate -j ro
```

```
Fileset Name Fileset Target                    Cache State Gateway Node Queue Length Queue numExec
------------ --------------                    ----------- ------------ ------------ -------------
ro           nfs://c26c3apv1/gpfs/homefs1/dir3 Active      c26c2apv2    0            7


List Policy:
RULE EXTERNAL LIST 'List' RULE 'List' LIST 'List' WHERE PATH_NAME LIKE '%'

Run the policy at home:
mmapplypolicy /gpfs/homefs1/dir3 -P px -f px.res -L 1 -N mount -I defer

Policy created this file, this should be hand-edited to retain only file names.

This file can be used at the cache to populate metadata.
```

```
# mmafmctl fs1 prefetch -j ro --metadata-only -list-file=px.res.list.List
mmafmctl: Performing prefetching of fileset: ro
     Queued    (Total)      Failed               TotalData
                                                 (approx in Bytes)
          0    (2)          0                    0
        100    (116)        5                    1368093971
        116    (116)        5                    1368093971

prefetch successfully queued at the gateway
```

```
Prefetch end can be monitored using this event:
Thu May 21 06:49:34.748 2015: [I] Calling User Exit Script prepop: event afmPrepopEnd,
Async command prepop.sh.


The statistics of the last prefetch command is viewed by:
```

```
# mmafmctl fs1 prefetch -j ro
Fileset Name Async Read (Pending) Async Read (Failed) Async Read (Already Cached) Async Read (Total) Async Read (Data
in Bytes)
------------ -------------------- ------------------  -------------------------- ------------------
-------------------------
ro           0                    1                   0                           7                0
```

8. **Prefetch of data using --home-list-file option:**

```
# cat /lfile1
/gpfs/homefs1/dir3/file1
/gpfs/homefs1/dir3/dir1/file1
```

```
# mmafmctl fs1 prefetch -j ro --home-list-file=/lfile1
mmafmctl: Performing prefetching of fileset: ro
Queued    (Total)      Failed          TotalData
                                       (approx in Bytes)
 0         (2)         0               0
 100       (116)       5               1368093971
 116       (116)       5               1368093971

prefetch successfully queued at the gateway
```

```
# mmafmctl fs1 prefetch -j ro
Fileset Name Async Read (Pending) Async Read (Failed) Async Read (Already Cached) Async Read (Total) Async Read (Data
in Bytes)
------------ -------------------- ------------------  -------------------------- ------------------
-------------------------
ro           0                    0                   0                          2                122880
```

9. **Prefetch of data using `--home-inode-file` option:**

```
Inode file is created using the above policy at home, and should be used as such without
hand-editing.

List Policy:
RULE EXTERNAL LIST 'List' RULE 'List' LIST 'List' WHERE PATH_NAME LIKE '%'
```

```
Run the policy at home:
# mmapplypolicy /gpfs/homefs1/dir3 -P px -f px.res -L 1 -N mount -I defer

# cat /lfile2
113289 65538 0 -- /gpfs/homefs1/dir3/file2
113292 65538 0 -- /gpfs/homefs1/dir3/dir1/file2
```

```
# mmafmctl fs1 prefetch -j ro2 --home-inode-file=/lfile2
mmafmctl: Performing prefetching of fileset: ro2
     Queued    (Total)      Failed         TotalData
                                           (approx in Bytes)
     0         (2)          0               0
     2         (2)          0               122880

prefetch successfully queued at the gateway
```

```
 mmafmctl fs1 prefetch -j ro
Fileset Name Async Read (Pending) Async Read (Failed) Async Read (Already Cached) Async Read (Total) Async Read (Data
in Bytes)
------------ -------------------- ------------------  -------------------------- ------------------
-------------------------
ro           0                    0                  2                          2                 0
```

10. **Using `--home-fs-path` option for a target with NSD protocol:**

```
# mmafmctl fs1 getstate -j ro2
Fileset Name Fileset Target              Cache State Gateway Node Queue Length Queue numExec
------------ --------------              ----------- ------------ ------------ -------------
ro2          gpfs:///gpfs/remotefs1/dir3 Active      c26c4apv1    0            7


# cat /lfile2
113289 65538 0 -- /gpfs/homefs1/dir3/file2
113292 65538 0 -- /gpfs/homefs1/dir3/dir1/file2
```

```
# mmafmctl fs1 prefetch -j ro2 --home-inode-file=/lfile2 --home-fs-path=/gpfs/homefs1/dir3
mmafmctl: Performing prefetching of fileset: ro2
Queued    (Total)      Failed          TotalData
                                       approx in Bytes)
0         (2)          0               0
2         (2)          0               122880

prefetch successfully queued at the gateway
```

```
# mmafmctl fs1 prefetch -j ro2
Fileset Name Async Read (Pending) Async Read (Failed) Async Read (Already Cached) Async Read (Total)
Async Read (Data in Bytes)
------------ -------------------- ------------------  -------------------------- ------------------
-------------------------
ro2          0                    0                  0                          2
122880
```

11. **Manually evicting using safe-limit and filename parameters:**

```
# ls -lis /gpfs/fs1/ro2/file10M_1
12605961 10240 -rw-r--r-- 1 root root 10485760 May 21 07:44 /gpfs/fs1/ro2/file10M_1

# mmafmctl fs1 evict -j ro2 --safe-limit=1 --filter FILENAME=file10M_1

# ls -lis /gpfs/fs1/ro2/file10M_1
12605961 0 -rw-r--r-- 1 root root 10485760 May 21 07:44 /gpfs/fs1/ro2/file10M_1
```

12. **IW Failback:**

```
# mmafmctl fs1 getstate -j iw1
Fileset Name Fileset Target                    Cache State Gateway Node Queue Length Queue numExec
```

```
------------ --------------                       ----------- ------------ ------------ -------------
iw1          nfs://c26c3apv1/gpfs/homefs1/dir3 Active      c25m4n03     0            8


# touch file3 file4

# mmafmctl fs1 getstate -j iw1
Fileset Name Fileset Target                    Cache State Gateway Node Queue Length Queue numExec
------------ --------------                    ----------- ------------ ------------ -------------
iw1          nfs://c26c3apv1/gpfs/homefs1/dir3 Dirty       c25m4n03     2            11


Unlink IW fileset feigning failure:
# mmunlinkfileset fs1 iw1 -f
Fileset iw1 unlinked.

Write from IW home, assuming applications failed over to home:
Thu May 21 08:20:41 4]dir3# touch file5 file6
Relink IW back on the cache cluster, assuming it came back up:
# mmlinkfileset fs1 iw1 -J /gpfs/fs1/iw1
Fileset iw1 linked at /gpfs/fs1/iw1

Run failback on IW:
# mmafmctl fs1 failback -j iw1 --start --failover-time='May 21 08:20:41'

# mmafmctl fs1 getstate -j iw1
Fileset Name Fileset Target                    Cache State    Gateway Node Queue Length Queue numExec
------------ --------------                    -----------    ------------ ------------ -------------
iw1          nfs://c26c3apv1/gpfs/homefs1/dir3 FailbackInProg c25m4n03     0            0

# mmafmctl fs1 failback -j iw1 -stop

# mmafmctl fs1 getstate -j iw1
Fileset Name Fileset Target                    Cache State Gateway Node Queue Length Queue numExec
------------ --------------                    ----------- ------------ ------------ -------------
iw1          nfs://c26c3apv1/gpfs/homefs1/dir3 Active      c25m4n03     0            3
```

13. **Manual evict using the --list-file option:**

```
# ls -lshi /gpfs/fs1/evictCache
total 6.0M
27858308 1.0M -rw-r--r--. 1 root root 1.0M Feb  5 02:07 file1M
27858307 2.0M -rw-r--r--. 1 root root 2.0M Feb  5 02:07 file2M
27858306 3.0M -rw-r--r--. 1 root root 3.0M Feb  5 02:07 file3M

# echo "RULE EXTERNAL LIST 'HomePREPDAEMON' RULE 'ListLargeFiles'
 LIST 'HomePREPDAEMON' WHERE PATH_NAME LIKE '%'" > /tmp/evictionPolicy.pol


# mmapplypolicy /gpfs/fs1/evictCache -I defer -P /tmp/evictionPolicy.pol
-f /tmp/evictionList

#Edited list of files to be evicted
[root@c21f2n08 ~]# cat /tmp/evictionList.list.HomePREPDAEMON
27858306 605742886 0   -- /gpfs/fs1/evictCache/file3M

# mmafmctl fs1 evict -j evictCache --list-file /tmp/evictionList.list.HomePREPDAEMON

# ls -lshi /gpfs/fs1/evictCache
total 3.0M
27858308 1.0M -rw-r--r--. 1 root root 1.0M Feb  5 02:07 file1M
27858307 2.0M -rw-r--r--. 1 root root 2.0M Feb  5 02:07 file2M
27858306    0 -rw-r--r--. 1 root root 3.0M Feb  5 02:07 file3M
```

14. **Manual evict using the `--file` option:**

```
# ls -lshi /gpfs/fs1/evictCache
total 3.0M
27858308 1.0M -rw-r--r--. 1 root root 1.0M Feb  5 02:07 file1M
27858307 2.0M -rw-r--r--. 1 root root 2.0M Feb  5 02:07 file2M
27858306    0 -rw-r--r--. 1 root root 3.0M Feb  5 02:07 file3M

# mmafmctl fs1 evict -j evictCache  --file /gpfs/fs1/evictCache/file1M

# ls -lshi /gpfs/fs1/evictCache
total 0
27858308 0 -rw-r--r--. 1 root root 1.0M Feb  5 02:07 file1M
27858307 0 -rw-r--r--. 1 root root 2.0M Feb  5 02:07 file2M
27858306 0 -rw-r--r--. 1 root root 3.0M Feb  5 02:07 file3M
```

See also

- “mmafmconfig command” on page 45
- “mmafmlocal command” on page 78
- “mmchattr command” on page 157
- “mmchconfig command” on page 170
- “mmchfileset command” on page 224
- “mmchfs command” on page 232
- “mmcrfileset command” on page 311
- “mmcrfs command” on page 318
- “mmlsconfig command” on page 495
- “mmlsfileset command” on page 501
- “mmlsfs command” on page 506
- “mmpsnap command” on page 614

See the AFM and AFM-based DR chapters in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide* for details.

Location

`/usr/lpp/mmfs/bin`

mmafmlocal command

Provides a list of cached files and file statistics such as inode number, allocated blocks, and so on.

Synopsis

```
mmafmlocal ls [FileName ...]
```

or

```
mmafmlocal stat FileName ...
```

Availability

Available on all IBM Spectrum Scale editions. Available on AIX and Linux.

Description

The `mmafmlocal` command provides information about files that exist in the cache.

Parameters

ls
Lists files with data that is in the cache already. This parameter is valid for fully-cached files only.

FileName
Specifies the name of a file to be listed.

stat
Displays statistics for files. If the file is not cached already, the number of allocated blocks is zero. This parameter is valid for partially-cached and fully-cached files.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmafmlocal` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To list information of all the cached files in a fileset:

   ```
   mmafmlocal ls
   ```

   The system displays information similar to:

   ```
   total 10240
   -rwxrwxrwx 1 root root 10485760 May 24 09:20 file1
   ```

2. To list information of a specific cached file in a fileset:

```
mmafmlocal ls file2
```

The system displays information similar to:

```
-rwxrwxrwx 1 root root 10485760 May 24 09:20 file2
```

3. To list the file statistics:

```
mmafmlocal stat file2
```

The system displays information similar to:

```
File: file2
   Inode number:  1582477
   Device ID:     0x2C (44)
   Size:          10485760
   Blocks:        20480
   Block size:    262144
   Links:         1
   Uid:           0 (root)
   Gid:           0 (root)
   Mode:          0100777
   Access time:   1464281093 (Thu May 26 16:44:53 2016 UTC)
   Modify time:   1464096038 (Tue May 24 13:20:38 2016 UTC)
   Change time:   1464096038 (Tue May 24 13:20:38 2016 UTC)
```

See also

- “mmafmconfig command” on page 45
- “mmafmctl command” on page 61
- “mmchattr command” on page 157
- “mmchconfig command” on page 170
- “mmchfileset command” on page 224
- “mmchfs command” on page 232
- “mmcrfileset command” on page 311
- “mmcrfs command” on page 318
- “mmlsconfig command” on page 495
- “mmlsfileset command” on page 501
- “mmlsfs command” on page 506
- “mmpsnap command” on page 614

Location

/usr/lpp/mmfs/bin

mmapplypolicy command

Deletes files, migrates files between storage pools, or does file compression or decompression in a file system as directed by policy rules.

Synopsis

```
mmapplypolicy {Device|Directory}
              [-A IscanBuckets] [-a IscanThreads] [-B MaxFiles]
              [-D yyyy-mm-dd[@hh:mm[:ss]]] [-e] [-f FileListPrefix]
              [-g GlobalWorkDirectory] [-I {yes|defer|test|prepare}]
              [-i InputFileList] [-L n] [-M name=value...] [-m ThreadLevel]
              [-N {all | mount | Node[,Node...] | NodeFile | NodeClass}]
              [-n DirThreadLevel] [-P PolicyFile] [-q] [-r FileListPathname...]
              [-S SnapshotName] [-s LocalWorkDirectory]
              [--choice-algorithm {best | exact | fast}]
              [--maxdepth MaxDirectoryDepth]
              [--max-merge-files MaxFiles] [--max-sort-bytes MaxBytes]
              [--other-sort-options SortOptions] [--qos QosClass]
              [--scope {filesystem | fileset | inodespace}]
              [--single-instance] [--sort-buffer-size Size]
              [--sort-command SortCommand] [--split-filelists-by-weight]
              [--split-margin n.n]
```

Availability

Available on all IBM Spectrum Scale editions. Available on AIX and Linux.

Description

You can use the `mmapplypolicy` command to apply rules that manage the following types of tasks:

- Migration and replication of file data to and from storage pools.
- Deleting files.
- File compression or decompression. For more information, see the topic *File compression* in the *IBM Spectrum Scale: Administration Guide*.

For more information about policy rules, see the topic *Policies for automating file management* in the *IBM Spectrum Scale: Administration Guide*.

Remember: When there are many millions of files to scan and process, the **mmapplypolicy** command can be very demanding of memory, CPU, I/O, and storage resources. Among the many options, consider specifying values for **-N**, **-g**, **-s**, **-a**, **-m**, **--sort-buffer-size**, and **--qos** to control the resource usage of **mmapplypolicy**.

You can run the `mmapplypolicy` command from any node in the cluster that has mounted the file system.

The `mmapplypolicy` command does not affect placement rules (for example, the SET POOL and RESTORE rule) that are installed for a file system by the `mmchpolicy` command. To display the currently installed rules, issue the `mmlspolicy` command.

A given file can match more than one list rule, but will be included in a given list only once. *ListName* provides the binding to an EXTERNAL LIST rule that specifies the executable program to use when processing the generated list.

The EXTERNAL POOL rule defines an external storage pool. This rule does not match files, but serves to define the binding between the policy language and the external storage manager that implements the external storage.

Any given file is a potential candidate for at most one MIGRATE or DELETE operation during one invocation of the `mmapplypolicy` command. That same file may also match the first applicable LIST rule.

A file that matches an EXCLUDE rule is not subject to any subsequent MIGRATE, DELETE, or LIST rules. You should carefully consider the order of rules within a policy to avoid unintended consequences.

For detailed information on GPFS policies, see the *IBM Spectrum Scale: Administration Guide*.

This command cannot be run from a Windows node. The GPFS API, documented functions in **gpfs.h** are not implemented on Windows, however the policy language does support the Windows file attributes, so you can manage your GPFS Windows files using the `mmapplypolicy` command running on an AIX or Linux node.

Note: To terminate `mmapplypolicy`, use the `kill` command to send a `SIGTERM` signal to the process group running `mmapplypolicy`.

For example, on Linux if you wanted to terminate `mmapplypolicy` on a process group whose ID is 3813, you would enter the following:

```
# kill -s SIGTERM -- -3813
```

If you need to determine which process group is running `mmapplypolicy`, you can use the following command (which also tells you which process groups are running `tsapolicy` and `mmhelp-apolicy`):

```
# mmdsh -N all ps auxw | grep policy
```

A sample output is as follows:

```
root     31666  0.0  0.0  84604  2328 ?        Sl   07:29   0:00 /usr/lpp/mmfs/bin/mmhelp-apolicy na -X
     10.222.4.12 -s /tmp -Y -x 36845 -m 24 -n 24 -a 2 -L 1 -d 00 -z 15
root      3813  0.3  0.1  68144  4792 pts/1    S    07:29   0:00 /bin/ksh /usr/lpp/mmfs/bin/mmapplypolicy
     /mak/millions -P /ghome/makaplan/policies/1p.policy -N all
root      3847  127  0.1 455228  5808 pts/1    Sl   07:29   0:38 /usr/lpp/mmfs/bin/tsapolicy /mak/
millions
     -P /ghome/makaplan/policies/1p.policy -I yes -L 1 -X 10.222.4.12 -N all
root      3850  0.0  0.0  84832  1620 pts/1    Sl   07:29   0:00 /usr/lpp/mmfs/bin/tsapolicy /mak/
millions
     -P /ghome/makaplan/policies/1p.policy -I yes -L 1 -X 10.222.4.12 -N all
root      3891  0.0  0.0  61156   768 pts/2    S+   07:29   0:00 grep policy
```

Parameters

Device
Specifies the device name of the file system from which files will have the policy rules applied. File system names need not be fully-qualified. `fs0` is just as acceptable as `/dev/fs0`. If specified, this must be the first parameter.

Directory
Specifies the fully-qualified path name of a GPFS file system subtree from which files will have the policy rules applied. If specified, this must be the first parameter.

-A *IscanBuckets*
Specifies the number of buckets of inode numbers (number of inode/filelists) to be created by the parallel directory scan and processed by the parallel inode scan. Affects the execution of the high-performance protocol that is used when both `-g` and `-N` are specified.

Tip: Set this parameter to the expected number of files to be scanned divided by one million. Then, each bucket will have about one million files.

-a *IscanThreads*
Specifies the number of threads and sort pipelines each node will run during the parallel inode scan and policy evaluation. It affects the execution of the high-performance protocol that is used when both `-g` and `-N` are specified. The default is 2. Using a moderately larger number can significantly improve performance, but might "strain" the resources of the node. In some environments a large value for this parameter can lead to a command failure.

Tip: Set this parameter to the number of CPU "cores" implemented on a typical node in your GPFS cluster.

-B *MaxFiles*
Specifies how many files are passed for each invocation of the EXEC script. The default value is 100.

If the number of files exceeds the value specified for *MaxFiles*, `mmapplypolicy` invokes the external program multiple times.

For more information about file list records, refer to the *IBM Spectrum Scale: Administration Guide*.

-D *yyyy-mm-dd*[*@hh:mm*[*:ss*]]
Specifies a date and optionally a (UTC) time as *year-month-day* at *hour:minute:second*.

The `mmapplypolicy` command evaluates policy rules as if it were running on the date and time specified by the `-D` flag. This can be useful for planning or testing policies, to see how the `mmapplypolicy` command would act in the future. If this flag is omitted, the `mmapplypolicy` command uses the current date and (UTC) time. If a date is specified but not a time, the time is assumed to be 00:00:00.

-e
Causes `mmapplypolicy` to re-evaluate and revalidate the following conditions immediately before executing the policy action for each chosen file:

- That the PATH_NAME still leads to the chosen file, and that the INODE and GENERATION values are the same.
- That the rule (iRule) still applies to, and is a first matching rule for, the chosen file.

Note: The `-e` option is particularly useful with `-r`, but can be used apart from it. It is useful because in the time that elapses after the policy evaluation and up to the policy execution, it is possible that the chosen pathname no longer refers to the same inode (for example the original file was removed or renamed), or that some of the attributes of the chosen file have changed in some way so that the chosen file no longer satisfies the conditions of the rule. In general, the longer the elapsed time, the more likely it is that conditions have changed (depending on how the file system is being used). For example, if files are only written once and never renamed or erased, except by policy rules that call for deletion after an expiration interval, then it is probably not necessary to re-evaluate with the `-e` option.

For more information about `-r`, see *IBM Spectrum Scale: Administration Guide*.

-f *FileListPrefix*
Specifies the location (a path name or file name prefix or directory) in which the file lists for external pool and list operations are stored when either the `-I defer` or `-I prepare` option is chosen. The default location is *LocalWorkDirectory*`/mmapplypolicy.`*processid*.

-g *GlobalWorkDirectory*
Specifies a global work directory in which one or more nodes can store temporary files during `mmapplypolicy` command processing. For more information about specifying more than one node to process the command, see the description of the `-N` option. For more information about temporary files, see the description of the `-s` option.

The global directory can be in the file system that `mmapplypolicy` is processing or in another file system. The file system must be a shared file system, and it must be mounted and available for reading and writing by every node that will participate in the `mmapplypolicy` command processing.

If the `-g` option is not specified, then the global work directory is the directory that is specified by the `sharedTmpDir` attribute of the `mmchconfig` command. For more information, see "mmchconfig command" on page 170. If the `sharedTmpDir` attribute is not set to a value, then the global work directory depends on the file system format version of the target file system:

- If the target file system is at file system format version 5.0.1 or later (file system format number 19.01 or later), then the global work directory is the directory `.mmSharedTmpDir` at the root level of the target file system.
- If the target file system is at a file system format version that is earlier than 5.0.1 then the command does not use a global work directory.

If the global work directory that is specified by `-g` option or by the `sharedTmpDir` attribute begins with a forward slash (/) then it is treated as an absolute path. Otherwise, it is treated as a path that is relative to the mount point of the file system.

If both the `-g` option and the `-s` option are specified, then temporary files can be stored in both the specified directories. In general, the local work directory contains temporary files that are written and read by a single node. The global work directory contains temporary files that are written and read by more than one node.

Note: The `mmapplypolicy` command uses high-performance, fault-tolerant protocols in its processing whenever both of the following conditions are true:

- The command is configured to run on multiple nodes.
- The command is configured to use a global work directory.

If the target file system is at file system format version 5.0.1 or later, then the command always uses high-performance, fault-tolerant protocols. The reason is that in this situation the command provides default values for running on multiple nodes (the node class `managerNodes`) and for sharing a global directory (`.mmSharedTmpDir`) if no explicit parameters are specified. For more information, see the following descriptions:

- The command runs on multiple nodes if any of the following circumstances are true:
  - The `-N` option on the command line specifies a set of helper nodes to run parallel instances of the policy code.
  - The `defaultHelperNodes` attribute of the `mmchconfig` command is set. This attribute specifies a list of helper nodes to be employed if the `-N` option is not specified.
  - The target file system is at file system format version 5.0.1 or later (file system format number 19.01 or later). If neither the `-N` option nor the `defaultHelperNodes` attribute is set, the members of the node class `managerNodes` are the helper nodes.
- The command uses a global work directory if any of the following circumstances are true:
  - The `-g` option on the command line specifies a global work directory.
  - The `sharedTmpDir` attribute of the `mmchconfig` command is set. This attribute specifies a global work directory to be used if the `-g` option is not specified.
  - The target file system is at file system format version 5.0.1 or later (file system format number 19.01 or later). If neither the `-g` option nor the `sharedTmpDir` attribute is set, the directory `.mmSharedTmpDir` at the root level of the target file system is the global work directory.

-I {yes | defer | test | prepare}
Specifies what actions the `mmapplypolicy` command performs on files:

yes
Indicates that all applicable policy rules are run, and the data movement between pools is done during the processing of the `mmapplypolicy` command. All defined external lists will be executed. This is the default action.

defer
Indicates that all applicable policy rules are run, but actual data movement between pools is deferred until the next `mmrestripefs` or `mmrestripefile` command. See also "-f FileListPrefix" on page 82.

test
Indicates that all policy rules are evaluated, but the `mmapplypolicy` command only displays the actions that would be performed had `-I defer` or `-I yes` been specified. There is no actual deletion of files or data movement between pools. This option is intended for testing the effects of particular policy rules.

prepare
Indicates that all policy execution is deferred and that `mmapplypolicy` only prepares file lists that are suitable for execution with the `–r` option. Records are written for each of the chosen files

and are stored in one or more file lists, under a path name that is specified by the `-f` option or in the default local work directory. The actual data movement occurs when the command is rerun with the `-r` option.

-i *InputFileList*
Specifies the path name for a user-provided input file list. This file list enables you to specify multiple starter directories or files. It can be in either of the following formats:

simple format file list
A list of records with the following format:

```
PATH_NAME
```

Each record represents either a single file or a directory. When a directory is specified, the command processes the entire subtree that is rooted at the specified path name.

File names can contain spaces and special characters; however, the special characters '\' and '\n' must be escaped with the '\' character similarly to the way `mmapplypolicy` writes path names in file lists for external pool and list operations.

The end-of-record character must be \n.

Example:

```
/mak/ea
/mak/old news
/mak/special\\stuff
```

`/usr/lpp/mmfs/samples/ilm/mmglobexpf.sample` is an example of a script that can be used to generate simple format file lists.

expert format file list
A list of records with the following format:

```
INODE:GENERATION:path-length!PATH_NAME end-of-record-character
```

Each record represents exactly one file.

The INODE and GENERATION values must be specified in hexadecimal format (`%11x`). If you do not know the generation number or inode number, specify `0` and GPFS will look it up for you.

The *path-length* value must be specified in decimal format (%d). The *path-length* value is followed by the delimiter `!`.

The end-of-record character must be \n or `\0`.

Example (the end-of-record characters are invisible):

```
00009a00:0:8!d14/f681
00009a01:1002:8!d14/f682
```

When you use an expert format file list, the directory scan phase is skipped and only the files that are specified with the *InputFileList* parameter are tested against the policy rules.

For more information, see the *IBM Spectrum Scale: Administration Guide*.

With either format, if a path name is not fully qualified, it is assumed to be relative to one of the following:

- the *Directory* parameter on the `mmapplypolicy` command invocation

Or

- the mount point of the GPFS file system, if *Device* is specified as the first argument

-L *n*
Controls the level of information displayed by the `mmapplypolicy` command. Larger values indicate the display of more detailed information. These terms are used:

candidate file
A file that matches a MIGRATE, DELETE, or LIST policy rule.

chosen file
A candidate file that has been scheduled for action.

These are the valid values for *n*:

0
Displays only serious errors.

1
Displays some information as the command runs, but not for each file. This is the default.

2
Displays each chosen file and the scheduled migration or deletion action.

3
Displays the same information as 2, plus each candidate file and the applicable rule.

4
Displays the same information as 3, plus each explicitly EXCLUDEed or LISTed file, and the applicable rule.

5
Displays the same information as 4, plus the attributes of candidate and EXCLUDEed or LISTed files.

6
Displays the same information as 5, plus non-candidate files and their attributes.

For examples and more information on this flag, see the section: *The mmapplypolicy -L command* in the *IBM Spectrum Scale: Problem Determination Guide*.

-M *name=value...*
Indicates a string substitution that will be made in the text of the policy rules before the rules are interpreted. This allows the administrator to reuse a single policy rule file for incremental backups without editing the file for each backup.

-m *ThreadLevel*
The number of threads that are created and dispatched within each mmapplypolicy process during the policy execution phase. The default value is 24.

-N {all | mount | *Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies a set of nodes to run parallel instances of policy code for better performance. The nodes must be in the same cluster as the node from which the mmapplypolicy command is issued. All node classes are supported. For more information about these options, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

If the -N option is not specified, then the command runs parallel instances of the policy code on the nodes that are specified by the defaultHelperNodes attribute of the mmchconfig command. For more information, see the topic "mmchconfig command" on page 170. If the defaultHelperNodes attribute is not set, then the list of helper nodes depends on the file system format version of the target file system. If the target file system is at file system format version 5.0.1 or later (file system format number 19.01 or later), then the helper nodes are the members of the node class managerNodes. Otherwise, the command runs only on the node where the mmapplypolicy command is issued.

-n *DirThreadLevel...*
The number of threads that will be created and dispatched within each mmapplypolicy process during the directory scan phase. The default is 24.

-P *PolicyFile*
Specifies the name of the file containing the policy rules to be applied. If not specified, the policy rules currently in effect for the file system are used. Use the mmlspolicy command to display the current policy rules.

-q
When specified, `mmapplypolicy` dispatches bunches of files from the file lists specified by the `-r` option in a round-robin fashion, so that the multithreaded (`-m`) and node parallel (`-N`) policy execution works on all the file lists "at the same time." When `-q` is not specified, policy execution works on the file lists sequentially. In either case bunches of files are dispatched for parallel execution to multiple threads (`-m`) on each of the possibly multiple nodes (`-N`).

-r *FileListPathname...*
Specifies one or more file lists of files for policy execution. The file lists that were used as input for `-r` were created by issuing `mmapplypolicy` with the `-I prepare` flag. You can specify several file lists by doing one of the following:

- Provide the path name of a directory of file lists, **or**
- Specify the `-r` option several times, each time with the path name of a different file list.

You can use this parameter to logically continue where `mmapplypolicy` left off when you specified the `-I prepare` option. To do this, invoke `mmapplypolicy` with all the same parameters and options (except the **-I prepare** option), and now substitute the `-r` option for `-f`. In between the invocations, you can process, reorder, filter, or edit the file lists that were created when you invoked `-I prepare`. You can specify any or all of the resulting file lists with the `-r` option.

The format of the records in each file list file can be expressed as:

```
iAggregate:WEIGHT:INODE:GENERATION:SIZE:iRule:resourceId:attr_flags:
path-lengthl!PATH_NAME:pool-length!POOL_NAME
[;show-length>!SHOW]end-of-record-character
```

For more information about file list records, refer to the *IBM Spectrum Scale: Administration Guide*.

-S *SnapshotName*
Specifies the name of a global snapshot for file system backup operations or for migrating snapshot data. The name appears as a subdirectory of the **.snapshots** directory in the file system root and can be found with the `mmlssnapshot` command.

Note: GPFS snapshots are read-only. Do not use deletion rules with `-S` *SnapshotName*.

Note: Do not use the `-S` option to scan files in a fileset snapshot. Instead, specify the full path of a directory in the snapshot as the first parameter of the command, as in the following example:

```
# mmapplypolicy <directory_path> -P policyfile ...
```

-s *LocalWorkDirectory*
Specifies a local directory in which one or more nodes can store temporary files during `mmapplypolicy` command processing. The default local work directory is `/tmp`. For more information about specifying more than one node to process the command, see the `-N` option of `mmapplypolicy`.

The temporary files contain lists of candidate files and lists of files that are selected to be processed. If the file system or directory that `mmapplypolicy` is processing contains many files, then the temporary files can occupy a large amount of storage space. To make a rough estimate of the size of the storage that is required, apply the formula `K * AVLP * NF`, where:

K
Is 3.75.

AVLP
Is the average length of the full path name of a file.

NF
Is the number of files that the command will process.

For example, if AVLP is 80, then the storage space that is required is roughly (300 * NF) bytes of temporary space.

--choice-algorithm {best | exact | fast}
Specifies one of the following types of algorithms that the policy engine is to use when selecting candidate files:

best
Chooses the optimal method based on the rest of the input parameters.

exact
Sorts all of the candidate files completely by weight, then serially considers each file from highest weight to lowest weight, choosing feasible candidates for migration, deletion, or listing according to any applicable rule LIMITs and current storage-pool occupancy. This is the default.

fast
Works together with the parallelized `-g /shared-tmp -N` node-list selection method. The `fast` choice method does not completely sort the candidates by weight. It uses a combination of statistical, heuristic, and parallel computing methods to favor higher weight candidate files over those of lower weight, but the set of chosen candidates may be somewhat different than those of the `exact` method, and the order in which the candidates are migrated, deleted, or listed is somewhat more random. The `fast` method uses statistics gathered during the policy evaluation phase. The `fast` choice method is especially fast when the collected statistics indicate that either all or none of the candidates are feasible.

--maxdepth ***MaxDirectoryDepth***
Specifies how deeply in the hierarchy of the starting directory to apply policies to files. If this parameter is omitted, the command applies the policies to all the subdirectories of the starting directory. A value of 0 causes the policies to be applied only to the files in the starting directory.

--max-merge-files ***MaxFiles***
Specifies the maximum number of files to be passed as input to the `sort` command for sorting and merging.

The `mmapplypolicy` command must do multiple, potentially large sorts and merges of temporary files as part of its processing. The `--max-merge-files` parameter specifies the maximum number of files that the `mmapplypolicy` command passes as input to a single `sort` command for sorting and merging.

If you set this value too high, the result can be excessive memory usage. The operating system can respond by terminating some of the `sort` processes, which causes the `mmapplypolicy` command to run for a longer time or to return with an error.

The default value of this parameter is 12. In general, it is a good practice to accept the default value of this parameter or to test carefully if you specify an overriding value.

See the related parameters `--max-sort-bytes` and `--sort-buffer-size`.

--max-sort-bytes ***MaxBytes***
Specifies the maximum number of bytes to be passed as input files to the `sort` command. This parameter does not apply to merges in which each of the input files has already been sorted.

The `mmapplypolicy` command must do multiple, potentially large sorts and merges of temporary files as part of its processing. The `--max-sort-bytes` parameter specifies the maximum number of bytes that the `mmapplypolicy` command can pass to a single instance of the `sort` command in one or more files.

If you set this value too high, the result can be excessive memory usage. The operating system can respond by terminating some of the `sort` processes, which causes the `mmapplypolicy` command to run for a longer time or to return with an error.

The default value of this parameter is 411 MB. In general, it is a good practice to accept the default value of this parameter or to test carefully if you specify an overriding value.

See the related parameters `--max-merge-files` and `--sort-buffer-size`.

--scope {filesystem | inodespace | fileset}
If *Device* is specified, the directory traversal starts at the root of the file system. `--scope` indicates one of the following levels of scope to be applied to the policy scan:

filesystem
The scan will involve the objects in the entire file system subtree pointed to by the *Directory* parameter. This is the default.

fileset
The scope of the scan is limited to the objects in the same fileset as the directory pointed to by the *Directory* parameter.

inodespace
The scope is limited to objects in the same single inode space from which the directory pointed to by the *Directory* parameter is allocated. The scan may span more than one fileset, if those filesets share the same inode space.

--qos *QosClass*
Specifies the Quality of Service for I/O operations (QoS) class to which the instance of the command is assigned. If you do not specify this parameter, the instance of the command is assigned by default to the `maintenance` QoS class. This parameter has no effect unless the QoS service is enabled. For more information, see the topic "mmchqos command" on page 263. Specify one of the following QoS classes:

`maintenance`
This QoS class is typically configured to have a smaller share of file system IOPS. Use this class for I/O-intensive, potentially long-running GPFS commands, so that they contribute less to reducing overall file system performance.

other
This QoS class is typically configured to have a larger share of file system IOPS. Use this class for administration commands that are not I/O-intensive.

For more information, see the topic *Setting the Quality of Service for I/O operations (QoS)* in *IBM Spectrum Scale: Administration Guide*.

--other-sort-options *SortOptions*
Passes options to the sort command (either the default sort command provided with the operating system, or an alternative sort command specified by the `--sort-command` parameter).

--single-instance
Ensures that, for the specified file system, only one instance of `mmapplypolicy` invoked with the `--single-instance` option can execute at one time. If another instance of `mmapplypolicy` invoked with the `--single-instance` option is currently executing, this invocation will do nothing but terminate.

--sort-buffer-size *Size*
Sets the sort-buffer size that is passed to the `sort` command. This parameter limits memory usage by the `sort` commands that the `mmapplypolicy` command calls to do sorts and merges.

The `mmapplypolicy` command must do multiple, potentially large sorts and merges of temporary files as part of its processing. It calls the operating-system `sort` command each time that it must do a sort. It can have multiple instances of the `sort` command running at the same time. If the numbers of items to be sorted are very large, the result can be excessive memory usage. The operating system can respond by terminating some of the sort processes, which causes the `mmapplypolicy` command to run for a longer time or to return with an error.

To prevent excessive memory consumption, you can set the `--sort-buffer-size` parameter to a lower value than its default. The `--sort-buffer-size` parameter is the value that the `mmapplypolicy` command passes to the `sort` command in the `buffer-size` parameter. The default value is 8%. If you need a lower value, you might set it to 5%.

You can specify the `--sort-buffer-size` parameter in any format that the `sort` program's `buffer-size` parameter accepts, such as "5%" or "1M".

In general, accept the default value of this parameter unless the system has excessive memory consumption that is attributable to large sort operations by the `mmapplypolicy` command.

See the related parameters `--max-merge-files` and `max-sort-bytes`.

--sort-command *SortCommand*
Specifies the fully-qualified path name for a POSIX-compliant sort command to be used instead of the default, standard command provided by the operating system.

Before specifying an alternative sort command (and for information about a suggested sort command), see *Improving performance with the --sort-command parameter* in *IBM Spectrum Scale: Administration Guide*.

--split-filelists-by-weight
Specifies that each of the generated file lists contain elements with the same `WEIGHT` value. This can be useful in conjunction with the `LIST` rule and the `WEIGHT (DIRECTORY_HASH)` clause. In this case, each generated list will contain files from the same directory.

Note: If you want all of the files from a given directory to appear in just one list, you might have to specify a sufficiently large `-B` value.

--split-margin *n.n*
A floating-point number that specifies the percentage within which the `fast`-choice algorithm is allowed to deviate from the `LIMIT` and `THRESHOLD` targets specified by the policy rules. For example if you specified a `THRESHOLD` number of 80% and a split-margin value of 0.2, the `fast`-choice algorithm could finish choosing files when it reached 80.2%, or it might choose files that bring the occupancy down to 79.8%. A nonzero value for split-margin can greatly accelerate the execution of the `fast`-choice algorithm when there are many small files. The default is 0.2.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmapplypolicy` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. This command displays the actions that would occur if a policy were applied, but does not apply the policy at this time:

```
# mmapplypolicy fs1 -P policyfile -I test
```

A sample output is as follows:

```
[I] GPFS current data pool utilization in KB and %
sp1     9728    19531264        0.049807%
sp2     4608    19531264        0.023593%
system  105216  19531264    0.538706%
[I] Loaded policy rules from fs1.pol.
Evaluating MIGRATE/DELETE/EXCLUDE rules with CURRENT_TIMESTAMP = 2009-02-27@20:00:22 UTC
parsed 2 Placement Rules, 0 Restore Rules, 3 Migrate/Delete/Exclude Rules,
        0 List Rules, 0 External Pool/List Rules
RULE 'sp1' SET POOL 'sp1' WHERE name like '%.sp1' or name like '%.tmp'
RULE 'default' SET POOL 'system'

RULE 'exclude *.save files' EXCLUDE WHERE NAME LIKE '%.save'

/* Deletion rule */
RULE 'delete' DELETE FROM POOL 'sp1' WHERE NAME LIKE '%.tmp'

/* Migration rule */
```

```
RULE 'migration to system pool' MIGRATE FROM POOL 'sp1' TO POOL 'system' WHERE NAME LIKE
'%sp1%'
[I] Directories scan: 11 files, 1 directories, 0 other objects, 0 'skipped' files and/or
errors.
[I] Inodes scan: 11 files, 1 directories, 0 other objects, 0 'skipped' files and/or errors.
[I] Summary of Rule Applicability and File Choices:
 Rule#  Hit_Cnt KB_Hit  Chosen  KB_Chosen       KB_Ill  Rule
  0     3       1536    0       0       0       RULE 'exclude *.save files' EXCLUDE WHERE(.)
  1     3       1536    3       1536    0       RULE 'delete' DELETE FROM POOL 'sp1' WHERE(.)
  2     2       1024    2       1024    0       RULE 'migration to system pool' MIGRATE FROM
POOL \
                                                'sp1' TO POOL 'system' WHERE(.)

[I] Files with no applicable rules: 4.

[I] GPFS Policy Decisions and File Choice Totals:
 Chose to migrate 1024KB: 2 of 2 candidates;
 Chose to premigrate 0KB: 0 candidates;
 Already co-managed 0KB: 0 candidates;
 Chose to delete 1536KB: 3 of 3 candidates;
 Chose to list 0KB: 0 of 0 candidates;
 0KB of chosen data is illplaced or illreplicated;
Predicted Data Pool Utilization in KB and %:
sp1     7168    19531264        0.036700%
sp2     4608    19531264        0.023593%
system  106240  19531264        0.543948%
```

2. This command applies a policy immediately:

```
# mmapplypolicy fs1 -P policyfile
```

A sample output is as follows:

```
[I] GPFS current data pool utilization in KB and %
sp1     9728    19531264        0.049807%
sp2     4608    19531264        0.023593%
system  105216  19531264        0.538706%
[I] Loaded policy rules from fs1.pol.
Evaluating MIGRATE/DELETE/EXCLUDE rules with CURRENT_TIMESTAMP = 2009-02-27@20:2
5:34 UTC
parsed 2 Placement Rules, 0 Restore Rules, 3 Migrate/Delete/Exclude Rules,
        0 List Rules, 0 External Pool/List Rules
RULE 'sp1' SET POOL 'sp1' WHERE name like '%.sp1' or name like '%.tmp'
RULE 'default' SET POOL 'system'

RULE 'exclude *.save files' EXCLUDE WHERE NAME LIKE '%.save'

/* Deletion rule */
RULE 'delete' DELETE FROM POOL 'sp1' WHERE NAME LIKE '%.tmp'

/* Migration rule */
RULE 'migration to system pool' MIGRATE FROM POOL 'sp1' TO POOL 'system' WHERE NAME LIKE
'%sp1%'
[I] Directories scan: 11 files, 1 directories, 0 other objects, 0 'skipped' files and/or
errors.
[I] Inodes scan: 11 files, 1 directories, 0 other objects, 0 'skipped' files and/or errors.
[I] Summary of Rule Applicability and File Choices:
 Rule#  Hit_Cnt KB_Hit  Chosen  KB_Chosen       KB_Ill  Rule
  0     3       3072    0       0       0       RULE 'exclude *.save files' EXCLUDE WHERE(.)
  1     3       3072    3       3072    0       RULE 'delete' DELETE FROM POOL 'sp1' WHERE(.)
  2     2       2048    2       2048    0       RULE 'migration to system pool'MIGRATE FROM
POOL \
                                                'sp1' TO POOL 'system' WHERE(.)

[I] Files with no applicable rules: 4.

[I] GPFS Policy Decisions and File Choice Totals:
 Chose to migrate 2048KB: 2 of 2 candidates;
 Chose to premigrate 0KB: 0 candidates;
 Already co-managed 0KB: 0 candidates;
 Chose to delete 3072KB: 3 of 3 candidates;
 Chose to list 0KB: 0 of 0 candidates;
 0KB of chosen data is illplaced or illreplicated;
Predicted Data Pool Utilization in KB and %:
sp1     4608    19531264        0.023593%
sp2     4608    19531264        0.023593%
system  107264  19531264        0.549191%
[I] A total of 5 files have been migrated, deleted or processed by an EXTERNAL E
```

```
XEC/script;
        0 'skipped' files and/or errors.
```

Additional examples of GPFS policies and using the `mmapplypolicy` command are in the *IBM Spectrum Scale: Administration Guide*.

See also

- “mmchpolicy command” on page 258
- “mmcrsnapshot command” on page 340
- “mmlspolicy command” on page 526
- “mmlssnapshot command” on page 540
- “mmsnapdir command” on page 722

Location

`/usr/lpp/mmfs/bin`

mmaudit command

Manages setting and viewing the file audit logging configuration in IBM Spectrum Scale.

Synopsis

```
mmaudit Device enable [--log-fileset FilesetName ]
                      [--retention Days] [--events {Event1[,Event2...] | ALL}]
                      [--compliant]
                      [{--filesets Fileset1[,Fileset2...] |
                      --skip-filesets Fileset1[,Fileset2...]}] [-q]
```

or

```
mmaudit Device disable [-q]
```

or

```
mmaudit Device update --events {Event1[,Event2...] | ALL} [-q]
```

or

```
mmaudit Device list [--events] [-Y]
```

or

```
mmaudit all list [--events] [-Y]
```

or

```
mmaudit all producerRestart -N { NodeName[,NodeName...] | NodeFile | NodeClass } [-q]
```

or

```
mmaudit all upgradePolicies
```

Availability

Available with IBM Spectrum Scale Advanced Edition, IBM Spectrum Scale Data Management Edition, IBM Spectrum Scale Developer Edition, or IBM Spectrum Scale Erasure Code Edition. Available on Linux x86 and Linux PPC LE.

Description

Enables, disables, and lists configuration data for file audit logging in a specified file system.
Lists all file audit logging enabled file systems in the cluster. Command messages are written to the `/var/adm/ras/mmaudit.log` file. The audit records are stored in the audit log fileset in a `/Device/.audit_log/audit_topic/Year/Month/Day` directory structure. The audit log files are named `auditLogFile_hostname_date_time`. The audit log files are rotated, compressed, and a retention date is set.

Note: When file audit logging is enabled on a file system, a fileset is created in the file system that is being audited. This fileset contains the audit logging files that contain the audit events. By default, this fileset is created as IAM mode noncompliant, but it can be created as IAM mode compliant if file audit logging is enabled with the **--compliant** option. By using either IAM mode, expiration dates are set for all files within the audit fileset. If the fileset is created in IAM mode noncompliant (the default), then the root user can change the expiration date to the current date so that audit files can be removed to free up disk space. If the fileset is created in IAM mode compliant (because of the use of the **--compliant** option), not even the root user can change the expiration dates of the audit logging files and they cannot be

removed until the expiration date. In addition, commands such as **mmrestorefs** will fail when restoring to a snapshot that would require removal of currently immutable (non-expired) files.

Parameters

Device
Specifies the device name of the file system upon which the audit log configuration change or listing is to occur.

all
Specifies that the command is executed against all devices configured for file audit logging. Currently, the only supported sub-commands are **list** and **upgradePolicies**.

enable
Enables file audit logging for the given device. Enablement entails setting up configuration and putting the audit policies in place.

The **--log-fileset** *FilesetName* option specifies the fileset name where the audit log records for the file system will be held. The default is `.audit_log`. The **--retention** *Days* option specifies the number of days to set the expiration date on all audit log record files when they are created. The default is 365 days. The **--events** option specifies the list of events that will be audited. For more information about the events that are supported, see *File audit logging events' descriptions* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*. The default is **ALL**. The **--degraded** option allows file audit logging to be enabled without as many default performance enhancements. The **--compliant** option specifies that the file audit logging fileset that is created to hold the file audit logging files will be IAM mode compliant. The default is noncompliant. In compliant mode, not even the root user can change the expiration dates of the file audit logging files to the past to free up space. The **--filesets** *Fileset1*[,*Fileset2*...] option specifies one or more filesets within the file system to audit. File system activity is only audited within these filesets and no other areas of the file system. The **--skip-filesets** *Fileset1*[,*Fileset2*...] option specifies one or more filesets within the file system not to audit. Audit events are generated for all file system activity except activity within this list of filesets.

disable
Disables file audit logging for the given device. Disablement removes audit policies and audit configurations that are specific to the device. Existing file audit records are changed to immutable and the retention period remains.

update
Updates the list of events that will be audited. The new event list will replace the existing set of events.

list --events [-Y]
Displays the file audit logging configuration information for the given device. The **all** option displays the file audit logging configuration information for all devices enabled for file audit logging. The **--events** option displays the device minor number, audit generation number, and a list of events that are being audited. The **-Y** option provides output in machine-readable (colon-delimited) format.

producerRestart -N { *NodeName*[,*NodeName*...] | *NodeFile* | *NodeClass* }
Restarts the producers for all file systems under audit on the nodes specified by the **-N** option. The **-N** option supports a comma-separated list of nodes, a full path name to a file containing node names, or a predefined node class.

Note: Issuing this command causes the event producers for clustered watch folder to be restarted as well.

upgradePolicies
Updates IBM Spectrum Scale policies that are associated with file audit logging enabled file systems to allow remotely mounted file systems to generate file audit logging events.

-q
Suppresses all **[I]** informational messages.

Exit status

0
Successful completion.

nonzero
A failure has occurred. Errors are written to `/var/adm/ras/mmaudit.log` and `/var/log/messages`.

Security

You must have root authority to run the **mmaudit** command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

Examples

1. To enable a file system with the default settings, issue the following command:

```
# mmaudit watch1 enable
[I] Successfully verified File Audit Logging type set to: FSYS
[I] Successfully updated File Audit Logging configuration for device: watch1
[I] Successfully checked or created File Audit Logging global catchall and config fileset
skip partitions for device: watch1
[I] Successfully created/linked File Audit Logging audit fileset .audit_log with link point /
watch1/.audit_log
[I] Successfully created File Audit Logging policy partition(s) to audit device: watch1
[I] Successfully enabled File Audit Logging for device: watch1
```

2. To enable a file system for a specific set of events, issue the following command:

```
# mmaudit watch1 enable --events OPEN,CLOSE
[I] Successfully verified File Audit Logging type set to: FSYS
[I] Successfully updated File Audit Logging configuration for device: watch1
[I] Successfully checked or created File Audit Logging global catchall and config fileset
skip partitions for device: watch1
[I] Successfully created/linked File Audit Logging audit fileset .audit_log with link point /
watch1/.audit_log
[I] Successfully created File Audit Logging policy partition(s) to audit device: watch1
[I] Successfully enabled File Audit Logging for device: watch1
```

3. To enable a file system with a different retention period, issue the following command:

```
# mmaudit watch1 enable --retention 90
[I] Successfully verified File Audit Logging type set to: FSYS
[I] Successfully updated File Audit Logging configuration for device: watch1
[I] Successfully checked or created File Audit Logging global catchall and config fileset
skip partitions for device: watch1
[I] Successfully created/linked File Audit Logging audit fileset .audit_log with link point /
watch1/.audit_log
[I] Successfully created File Audit Logging policy partition(s) to audit device: watch1
[I] Successfully enabled File Audit Logging for device: watch1
```

4. To specify one or more filesets to audit, issue the following command:

```
# mmaudit watch1 enable --log-fileset auditFset --retention 25 --filesets dep1,dep2,ind1,ind2
[I] Successfully verified filesets for File Audit Logging type: FILESET
[I] Successfully updated File Audit Logging configuration for device: watch1
[I] Successfully checked or created File Audit Logging global catchall and config fileset
skip partitions for device: watch1
[I] Successfully created/linked File Audit Logging audit fileset auditFset with link point /
watch1/auditFset
[I] Successfully created File Audit Logging policy partition(s) to audit device: watch1
[I] Successfully enabled File Audit Logging for device: watch1
```

5. To disable a file system that was previously enabled, issue the following command:

```
# mmaudit watch1 disable
```

```
[I] Successfully deleted File Audit Logging policy partition(s) for device: watch1
[I] Successfully updated File Audit Logging configuration for device: watch1
[I] Successfully checked or removed File Audit Logging global catchall and config fileset
skip partitions for device: watch1
[I] Successfully disabled File Audit Logging for device: watch1
```

6. To update the list of events that are being audited for a specific file system to available events, issue the following command:

```
# mmaudit fs3 update --events ALL
[I] Successfully updated the File Audit Logging policies for device fs3
```

7. To see which compliance type the file audit logging fileset is configured with, issue the following command:

```
# mmaudit all list -Y
mmaudit::HEADER:version:RESERVED:RESERVED:complianceType:auditDeviceName:cluster
ID:auditFilesetDeviceName:auditFilesetName:auditRetention:topicGenNum:eventTypes:
partitionMultiplier:auditType:filesets:
mmaudit:::3:::compliant:fs1:666656136840726581O:fs1:compliant:365:54:ACLCHANGE,
CLOSE,CREATE,GPFSATTRCHANGE,OPEN,RENAME,RMDIR,UNLINK,XATTRCHANGE:2:FSYS::
```

8. To see which file systems are currently configured for file audit logging, issue the following command:

```
# mmaudit all list
Audit              Cluster              Audit Fileset      Retention  Audit Type
Device             ID                   Name               (Days)     (Possible Filesets)
-------------------------------------------------------------------------------------------
fs0                11430652110915196903 john1              25         FILESET
                                                                       dep1,dep2,ind1,ind2
fs1                11430652110915196903 john2              75         SKIPFILESET
                                                                       dep1,dep2,ind1,ind2
fs2                11430652110915196903 john3              25         FSYS
```

9. To see which events are currently enabled for a file system, issue the following command:

```
# mmaudit fs3 list --events

Audit       Device    Audit     Event
Device      Minor     Gen       Types
-------------------------------------------------------------------------------------------
fs3         152       7         CLOSE,OPEN
```

See also

- “mmwatch command” on page 764

Location

/usr/lpp/mmfs/bin

mmauth command

Manages secure access to GPFS file systems.

Synopsis

```
mmauth genkey {new | commit | propagate [-N {Node[,Node...] | NodeFile | NodeClass}]}
```

or

```
mmauth add RemoteClusterName -k KeyFile [-l CipherList]
```

or

```
mmauth update RemoteClusterName {[-C NewClusterName] [-k KeyFile] [-l CipherList]}
```

or

```
mmauth delete {RemoteClusterName | all}
```

or

```
mmauth grant {RemoteClusterName | all} -f {Device | all} [-a {rw | ro}] [-r {uid:gid | no}]
```

or

```
mmauth deny {RemoteClusterName | all} -f {Device | all}
```

or

```
mmauth show [RemoteClusterName | all | ciphers] [-Y]
```

or

```
mmauth gencert --cname CanonicalName --cert ServerCertificateFile --out OutputKeystoreFile
               --label ClientCertificateLabel [--pwd-file KeystorePasswordFile]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmauth` command prepares a cluster to grant secure access to file systems owned locally. The `mmauth` command also prepares a cluster to receive secure access to file systems owned by another cluster. Use the `mmauth` command to generate a public/private key pair for the local cluster. A public/private key pair must be generated on both the cluster owning the file system and the cluster desiring access to the file system. The administrators of the clusters are responsible for exchanging the public portion of the public/private key pair. Use the `mmauth` command to add or delete permission for a cluster to mount file systems owned by the local cluster.

When a cluster generates a new public/private key pair, administrators of clusters participating in remote file system mounts are responsible for exchanging their respective public key file `/var/mmfs/ssl/id_rsa.pub` generated by this command.

The administrator of a cluster desiring to mount a file system from another cluster must provide the received key file as input to the `mmremotecluster` command. The administrator of a cluster allowing another cluster to mount a file system must provide the received key file to the `mmauth` command.

The keyword appearing after `mmauth` determines which action is performed:

add
Adds a cluster and its associated public key to the list of clusters authorized to connect to this cluster for the purpose of mounting file systems owned by this cluster.

delete
Deletes a cluster and its associated public key from the list of clusters authorized to mount file systems owned by this cluster.

deny
Denies a cluster the authority to mount a specific file system owned by this cluster.

gencert
Creates a client keystore with the keys and certificates required to communicate with the ISKLM key server.

genkey
Controls the generation and propagation of the OpenSSL key files:

new
Generates a new public/private key pair for this cluster. The key pair is placed in `/var/mmfs/ssl`. This must be done at least once before `cipherList`, the GPFS configuration parameter that enables GPFS with OpenSSL, is set.

The new key is in addition to the currently in effect committed key. Both keys are accepted until the administrator runs `mmauth genkey commit`.

commit
Commits the new public/private key pair for this cluster. Once `mmauth genkey commit` is run, the old key pair will no longer be accepted, and remote clusters that have not updated their keys (by running `mmauth update` or `mmremotecluster update`) will be disconnected.

propagate
Ensures that the currently in effect key files are placed in **/var/mmfs/ssl** on the nodes specified with the `-N` parameter. This may be necessary if the key files are lost and `adminMode central` is in effect for the cluster.

grant
Allows a cluster to mount a specific file system owned by this cluster.

show
Shows the list of clusters authorized to mount file system owned by this cluster.

update
Updates the public key and other information associated with a cluster authorized to mount file systems owned by this cluster.

When the local cluster name (or '.') is specified, `mmauth update -l` can be used to set the *cipherList* value for the local cluster. Note that you cannot use this command to change the name of the local cluster. Use the `mmchcluster` command for this purpose.

Note: Access controls such as grant and deny do not apply on the remote nodes that have the specified file system currently mounted. The new access control will be applied on the next mount of the file system.

Parameters

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies the nodes on which the key files should be restored. The default is `-N all`.

For general information on how to specify node names, see *Specifying nodes as input to GPFS(tm) commands* in *IBM Spectrum Scale: Administration Guide*.

This command does not support a *NodeClass* of `mount`.

RemoteClusterName
Specifies the remote cluster name requesting access to local GPFS file systems.

all
Indicates all remote clusters defined to the local cluster.

ciphers
Shows the supported ciphers.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

Options

-a {rw | ro}
Specifies the type of access allowed:

ro
Specifies read-only access.

rw
Specifies read/write access. This is the default.

-C *NewClusterName*
Specifies a new, fully-qualified cluster name for the already-defined cluster *RemoteClusterName*.

-f {*Device* | all}
Specifies the device name for a file system owned by this cluster. The *Device* argument is required. If `all` is specified, the command applies to all file systems owned by this cluster at the time that the command is issued.

-k *KeyFile*
Specifies the public key file generated by the `mmauth` command in the cluster requesting to remotely mount the local GPFS file system.

-l *CipherList*
Sets the security mode for communications between the current cluster and the remote cluster that is specified in the *RemoteClusterName* parameter. There are three security modes:

EMPTY
The sending node and the receiving node do not authenticate each other, do not encrypt transmitted data, and do not check data integrity.

AUTHONLY
The sending and receiving nodes authenticate each other, but they do not encrypt transmitted data and do not check data integrity. This mode is the default in IBM Spectrum Scale V4.2 or later.

Cipher
The sending and receiving nodes authenticate each other, encrypt transmitted data, and check data integrity. To set this mode, you must specify the name of a supported cipher, such as `AES128-GCM-SHA256`.

For more information about the security mode and supported ciphers, see the topic *Security mode* in the *IBM Spectrum Scale: Administration Guide*.

-r {*uid*:*gid* | no}
Specifies a root credentials remapping (*root squash*) option. The UID and GID of all processes with root credentials from the remote cluster will be remapped to the specified values. The default is not to remap the root UID and GID. The *uid* and *gid* must be specified as unsigned integers or as symbolic names that can be resolved by the operating system to a valid UID and GID. Specifying `no`, `off`, or `DEFAULT` turns off the remapping.

For more information, see the *IBM Spectrum Scale: Administration Guide* and search on *root squash*.

--cname *CanonicalName*
Specifies the canonical name of the client used in the certificate.

--cert *ServerCertificateFile*
Specifies the path name to a file containing an ISKLM certificate.

--out *OutputKeystoreFile*
Specifies the path name for the file that will contain the keystore.

--pwd-file *KeystorePasswordFile*
Specifies the keystore password file. If omitted, you will be prompted to enter the keystore password. A maximum of 20 characters are allowed. Only the following characters are allowed in the password:

`0 to 9`

`A to Z`

`a to z`

`!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~`

The `--pwd` *KeystorePassword* option is considered deprecated and may be removed in a future release.

--label *ClientCertificateLabel*
Specifies the label of the client certificate within the keystore. A maximum of 20 characters are allowed.

Exit status

0
Successful completion. After a successful completion of the `mmauth` command, the configuration change request will have been propagated to all nodes in the cluster.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmauth` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. This is an example of an `mmauth genkey new` command:

```
# mmauth genkey new
```

   A sample output is as follows:

```
Generating RSA private key, 512 bit long modulus
............+++++++++++++.+++++++++++++
e is 65537 (0x10001)
mmauth: Command successfully completed
mmauth: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

2. This is an example of an `mmauth genkey commit` command:

```
# mmauth genkey commit
```

A sample output is as follows:

```
mmauth: Command successfully completed
mmauth: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

3. This is an example of an `mmauth add` command:

```
# mmauth add clustA.kgn.ibm.com -k /u/admin/keys/clustA.pub
```

A sample output is as follows:

```
mmauth: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

4. This is an example of an `mmauth update` command:

```
# mmauth update clustA.kgn.ibm.com -k /u/admin/keys/clustA_new.pub
```

A sample output is as follows:

```
mmauth: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

5. This is an example of an `mmauth grant` command:

```
# mmauth grant clustA.kgn.ibm.com -f /dev/gpfs1 -a ro
```

A sample output is as follows:

```
mmauth: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

6. This is an example of an `mmauth show` command:

```
# mmauth show all
```

A sample output is as follows:

```
Cluster name:        clustA.kgn.ibm.com
Cipher list:         AES128-SHA
SHA digest:          a3917c8282fca7a27d951566940768dcd241902b
File system access:  gpfs1 (ro)
```

```
Cluster name:        clustB.kgn.ibm.com (this cluster)
Cipher list:         AES128-SHA
SHA digest:          6ba5e3c1038246fe30f3fc8c1181fbb2130d7a8a
SHA digest (new):    3c1038246fe30f3fc8c1181fbb2130d7a8a9ab4d
File system access:  (all rw)
```

For `clustB.kgn.ibm.com`, the `mmauth genkey new` command has been issued, but the `mmauth genkey commit` command has not yet been issued.

For more information on the SHA digest, see *The SHA digest* in *IBM Spectrum Scale: Problem Determination Guide*.

See also

- “mmremotefs command” on page 663
- “mmremotecluster command” on page 660

See also the topic about accessing GPFS file systems from other GPFS clusters in the *IBM Spectrum Scale: Administration Guide*.

Location

`/usr/lpp/mmfs/bin`

mmbackup command

Performs a backup of a GPFS file system or independent fileset to an IBM Spectrum Protect server.

Synopsis

```
mmbackup {Device | Directory} [-t {full | incremental}]
         [-N {Node[,Node...] | NodeFile | NodeClass}]
         [-g GlobalWorkDirectory] [-s LocalWorkDirectory]
         [-S SnapshotName] [-f] [-q] [-v] [-d]
         [-a IscanThreads] [-n DirThreadLevel]
         [-m ExecThreads | [[--expire-threads ExpireThreads] [--backup-threads BackupThreads |
         [[--selective-backup-threads selBackupThreads]
         [--incremental-backup-threads incBackupThreads]]]]]
         [-B MaxFiles | [[--max-backup-count MaxBackupCount |
         [[--max-incremental-backup-count MaxIncBackupCount]
         [--max-selective-backup-count MaxSelBackupCount]]]
         [--max-expire-count MaxExpireCount]]]
         [--max-backup-size MaxBackupSize] [--qos QosClass] [--quote | --noquote]
         [--rebuild] [--scope {filesystem | inodespace}]
         [--backup-migrated | --skip-migrated] [--tsm-servers TSMServer[,TSMServer...]]
         [--tsm-errorlog TSMErrorLogFile] [-L n] [-P PolicyFile]
```

Availability

Available on all IBM Spectrum Scale editions. Available on AIX and Linux.

Description

Use the **mmbackup** command to back up the user data from a GPFS file system or independent fileset to an IBM Spectrum Protect server or servers. The **mmbackup** command can be used only to back up file systems that are owned by the local cluster.

Attention: In IBM Spectrum Scale 4.1 and later, a full backup (`-t full`) with **mmbackup** is required if a full backup was never performed with GPFS 3.3 or later. For more information, see the topic *File systems backed up using GPFS 3.2 or earlier versions of mmbackup* in the *IBM Spectrum Scale: Administration Guide*.

The IBM Spectrum Protect Backup-Archive client must be installed and at the same version on all the nodes that are executing the **mmbackup** command or are named in a node specification with `-N`. For more information about IBM Spectrum Protect requirements for the **mmbackup** command, see the topic *Firewall recommendations for using IBM Spectrum Protect with IBM Spectrum Scale* in the *IBM Spectrum Scale: Administration Guide*.

You can run multiple simultaneous instances of **mmbackup** if they are on different file systems or filesets.

See the topic *Backup considerations for using IBM Spectrum Protect* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide* and the topic *Configuration reference for using IBM Spectrum Protect with IBM Spectrum Scale* in the *IBM Spectrum Scale: Administration Guide*.

Parameters

Device
The device name for the file system to be backed up. File system names do not need to be fully qualified. `fs0` is as acceptable as `/dev/fs0`.

Directory
Specifies either the mount point of a GPFS file system or the path to an independent fileset root to back up. The path `/gpfs/fs0` can be used to specify the GPFS file system called `fs0`.

Notes:

- Do not specify a subdirectory path. Doing so can result in inconsistent backups.
- Do not specify a snapshot directory path. To back up a snapshot, use `-S` *SnapshotName*.

- If you specify the path of an independent fileset root, you must also specify `--scope` *inodespace*.

-t {full | incremental}
Specifies whether to perform a full backup of all of the files in the file system, or an incremental backup of only those files that changed since the last backup was performed. The default is an `incremental` backup.

Note on GPFS 3.2: A full backup expires all GPFS 3.2 format TSM inventory if all previous backups were incremental and the GPFS 3.2 backup format was in use previously. If **mmbackup** on GPFS 3.2 was first used, and then only incremental backups were done on GPFS 3.4 or 3.5, then TSM still contains 3.2 format backup inventory. This inventory will automatically be marked for expiration by **mmbackup** after a successful full or incremental backup.

Note: Do not use `-t full` with an unlinked fileset. Use an EXCLUDE statement to exclude directories or files from the backup operation.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies a list of nodes to run parallel instances of the backup process for better performance. The IBM Spectrum Protect Backup-Archive client must be installed on each node in the list. All the nodes must be in the same cluster as the node from which the `mmbackup` command is issued. All node classes are supported. For more information about these options, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

If the `-N` option is not specified, then the command runs parallel instances of the backup process on the nodes that are specified by the `defaultHelperNodes` attribute of the `mmchconfig` command. If the `defaultHelperNodes` attribute is not set, then the command runs only on the node where the **mmbackup** command is issued. For more information about the `defaultHelperNodes` attribute, see the topic "mmchconfig command" on page 170.

-g *GlobalWorkDirectory*
Specifies a global work directory in which one or more nodes can store temporary files during `mmbackup` command processing. For more information about specifying more than one node to process the command, see the description of the `-N` option.

The global directory must be in a shared file system, and the file system must be mounted and available for reading and writing by every node that will participate in the `mmbackup` command processing.

If the `-g` option is not specified, then the global work directory is the directory that is specified by the `sharedTmpDir` attribute of the `mmchconfig` command. For more information, see "mmchconfig command" on page 170. If the `sharedTmpDir` attribute is not set to a value, then the global work directory depends on the file system format version of the target file system:

- If the target file system is at file system format version 5.0.1 or later (file system format number 19.01 or later), then the global work directory is the directory `.mmSharedTmpDir` at the root level of the target file system.
- If the target file system is at a file system format version that is earlier than 5.0.1, then the command does not use a global work directory.

If the global work directory that is specified by `-g` option or by the `sharedTmpDir` attribute begins with a forward slash (/) then it is treated as an absolute path. Otherwise it is treated as a path that is relative to the mount point of the file system.

If both the `-g` option and the `-s` option are specified, then temporary files can be stored in both the specified directories. In general, the local work directory contains temporary files that are written and read by a single node. The global work directory contains temporary files that are written and read by more than one node.

Note: The specified global work directory and its contents are excluded from the file system or fileset backup. If the directory is located in the file system that is being backed up, make sure that you use a directory that is dedicated to the working files. For example, if you specify the root directory of the file system, the entire contents of the file system are excluded.

-s ***LocalWorkDirectory***
Specifies the local directory to be used for temporary storage during **mmbackup** command processing. The default directory is `/tmp`. A *LocalWorkDirectory* must exist on each node that is used to run the **mmbackup** command or specified in a node specification with `-N`.

-S ***SnapshotName***
Specifies the name of a global snapshot for any backup operations or a fileset-level snapshot if `--scope inodespace` is also specified for a fileset backup. The snapshot must be created before the **mmbackup** command is used. Snapshot names can be found with the **mmlssnapshot** command. If a fileset is not present in the named snapshot and fileset-level backup is invoked with `--scope inodespace`, an error is returned. If a fileset-level snapshot name is used with a file system backup, an error is returned.

The use of `-S` *SnapshotName* is recommended because it provides **mmbackup** and IBM Spectrum Protect a consistent view of GPFS from which to perform backup. Deletion of the snapshot that is used for backup can be performed with the **mmdelsnapshot** command after **mmbackup** completes.

Note: The *SnapshotName* that is specified must be unique to the file system.

-f
Specifies that processing continues when unlinked filesets are detected. All files that belong to unlinked filesets are ignored.

Note: Because `-f` has a large impact on performance, avoid using it unless it is necessary to perform a backup operation with unlinked filesets.

-q
Performs a query operation before issuing **mmbackup**. The IBM Spectrum Protect server might have data stored already that is not recognized as having been backed up by **mmbackup** and its own shadow database. To properly compute the set of files that currently need to be backed up, **mmbackup** can perform an IBM Spectrum Protect query and process the results to update its shadow database. Use the `-q` switch to perform this query and then immediately commence the requested backup operation.

Note: Do not use `-q` with the `--rebuild` parameter.

-v
Specifies verbose message output. Use this flag to cause **mmbackup** to issue more verbose messages about its processing. See also “Environment” on page 106.

-d
Gathers debugging information that is useful to the IBM Support Center for diagnosing problems.

-a ***IscanThreads***
Specifies the number of threads and sort pipelines each node runs during parallel inode scan and policy evaluation. The default value is 2.

-n ***DirThreadLevel***
Specifies the number of threads that are created and dispatched within each **mmapplypolicy** process during the directory scan phase. The default value is 24.

-m ***ExecThreads***
Specifies the number of threads that are created and dispatched within each **mmapplypolicy** process during the policy execution phase. The default value for **mmapplypolicy** is 24; however, the default value for **mmbackup** is 1. This option cannot be used with the `--expire-threads`, `--backup-threads`, `--selective-backup-threads`, or `--incremental-backup-threads` options.

--expire-threads ***ExpireThreads***
Specifies the number of worker threads permitted on each node to perform `dsmc expire` tasks in parallel. This option cannot be used with the `-m` option. Valid values are 1 - 32. The default value is 4.

--backup-threads ***BackupThreads***
Specifies the number of worker threads that are permitted on each node to perform `dsmc selective` or `dsmc incremental` tasks in parallel. This option cannot be used with the `-m`,

`--incremental-backup-threads`, or `--selective-backup-threads` options. Valid values are 1 - 32. The default value is 1.

--selective-backup-threads *selBackupThreads*
Specifies the number of worker threads that are permitted on each node to perform `dsmc selective` tasks in parallel. This option cannot be used with the `-m` or `--backup-threads` options. Valid values are 1 - 32. The default value is 1.

--incremental-backup-threads *incBackupThreads*
Specifies the number of worker threads that are permitted on each node to perform `dsmc incremental` tasks in parallel. This option cannot be used with the `-m`, `--backup-threads`, or `-t full` options. Valid values are 1 - 32. The default value is 1.

-B *MaxFiles*
Specifies the maximum number of objects in a bunch for each invocation of the **mmapplypolicy** EXEC script. If this option is not specified, the ideal bunch count is automatically computed. This option cannot be used with the `--max-backup-count`, `--max-expire-count`, `--max-selective-backup-count`, or `--max-incremental-backup-count` options. Valid values are 100 - 2147483647.

--max-backup-count *MaxBackupCount*
Specifies the maximum number of objects in a bunch for each `dsmc selective` or `dsmc incremental` command. This option cannot be used with the `-B`, `--max-incremental-backup-count`, or `--max-selective-backup-count` options. Valid values are 100 - 2147483647.

--max-incremental-backup-count *MaxIncBackupCount*
Specifies a limit on the maximum number of objects in any single `dsmc incremental backup` command to be n objects in a bunch. This option cannot be used with the `-B`, `--max-backup-count`, or `-t full` options.

--max-selective-backup-count *MaxSelBackupCount*
Specifies a limit on the maximum number of objects in any single `dsmc selective backup` command to be n objects in a bunch. This option cannot be used with the `-B` or `--max-backup-count` options.

--max-expire-count *MaxExpireCount*
Specifies the maximum number of objects in a bunch for each `dsmc expire` command. This option cannot be used with the `-B` option. Valid values are 100 - 2147483647.

--max-backup-size *MaxBackupSize*
Specifies a policy limit on the content size in KB for each `dsmc selective` or `dsmc incremental` command.

--qos *QosClass*
Specifies the Quality of Service for I/O operations (QoS) class to which the instance of the command is assigned. If you do not specify this parameter, the instance of the command is assigned by default to the `maintenance` QoS class. This parameter has no effect unless the QoS service is enabled. For more information, see the topic "mmchqos command" on page 263. Specify one of the following QoS classes:

maintenance
This QoS class is typically configured to have a smaller share of file system IOPS. Use this class for I/O-intensive, potentially long-running GPFS commands, so that they contribute less to reducing overall file system performance.

other
This QoS class is typically configured to have a larger share of file system IOPS. Use this class for administration commands that are not I/O-intensive.

For more information, see the topic *Setting the Quality of Service for I/O operations (QoS)* in *IBM Spectrum Scale: Administration Guide*.

--quote | --noquote
Specifies whether to decorate (or not to decorate) file-list entries with quotation marks. The **mmbackup** command uses file lists to convey the lists of files to the IBM Spectrum Protect Backup-

Archive client program. Depending on IBM Spectrum Protect client configuration options, the file lists might not require each file name to be surrounded by quotation marks. If certain IBM Spectrum Protect client configuration options are in use, do not add quotation marks to file-list entries. Use the `--noquote` option in these instances.

--rebuild
Specifies whether to rebuild the **mmbackup** shadow database from the inventory of the IBM Spectrum Protect server. This option is similar to the `-q` option; however, no backup operation proceeds after the shadow database is rebuilt. Use this option if the shadow database of **mmbackup** is known to be out of date, but the rebuilding operation must be done at a time when the IBM Spectrum Protect server is less loaded than during the normal time **mmbackup** is run.

If backup files with the old snapshot name /*Device*/`.snapshots`/`.mmbuSnapshot` exist in the inventory of the IBM Spectrum Protect server, those files will be expired from the IBM Spectrum Protect server after the shadow database is rebuilt and any successful incremental or full backup completes.

Note: Do not use `--rebuild` with the `-q` parameter.

--scope {filesystem | inodespace}
Specifies that one of the following traversal scopes be applied to the policy scan and backup candidate selection:

filesystem
Scans all the objects in the file system that are specified by *Device* or that are mounted at the *Directory* specified. This is the default behavior.

inodespace
Specifies that the scan is limited in scope to objects in the same single inode space from which the *Directory* is allocated. The scan might span more than one fileset if those filesets share inode space; for example, dependent filesets.

--backup-migrated | --skip-migrated
HSM migrated objects are normally skipped even if they need to be backed up.

`--backup-migrated`

When specified, HSM migrated objects that need to be backed up are sent for backup. This can cause an HSM recall and other associated costs. When the same IBM Spectrum Protect server instance is used for HSM and backup, this option protects migrated, changed objects more efficiently. These objects are copied internally on the IBM Spectrum Protect server from the HSM storage pool directly to the BACKUP storage pool without incurring object recalls and their local storage costs. After backup, these objects will remain migrated. This occurs only if the object has no ACL or XAttrs attached.

`--skip-migrated`

When specified, HSM migrated objects that need to be backed up are not sent for backup. The path names of these objects are listed in a file, `mmbackup.hsmMigFiles.TSMServer`, that is located in the Directory or on the Device that is being backed up. The path names that are listed can then be recalled by the administrator in a staged manner. This is the default action that is taken for HSM migrated objects.

--tsm-servers *TSMServer*[,*TSMServer*...]
Specifies the name of the IBM Spectrum Protect server or servers that are to be used for this backup. Each of the servers is used for the specified backup task.

If this option is not specified and **MMBACKUP_SERVER_FROM_OPT** is set, the **mmbackup** command will use the server defined by `dsmc q opt SERVERNAME` or will backup to the servers that are specified in the `dsm.sys` file.

--tsm-errorlog *TSMErrorLogFile*
Specifies the path name of the log file to pass to IBM Spectrum Protect Backup-Archive client commands.

-L *n*
Controls the level of information that is displayed by the **mmapplypolicy** command that is invoked by using the **mmbackup** command. This option and its value are passed directly to the **mmapplypolicy** command (the use of this option is generally for debugging purposes). Refer to the **mmapplypolicy** command for an explanation of supported values.

-P *PolicyFile*
Specifies a customized policy rules file for the backup.

Environment

The behavior of **mmbackup** can be influenced by several environment variables when set.

Variables that apply to IBM Spectrum Protect Backup-Archive client program dsmc

MMBACKUP_DSMC_MISC
The value of this variable is passed as arguments to `dsmc restore` and `dsmc query {backup,inclexcl,session}` commands.

MMBACKUP_DSMC_BACKUP
The value of this variable is passed as arguments to `dsmc`, `dsmc selective`, and `dsmc incremental` commands.

MMBACKUP_DSMC_EXPIRE
The value of this variable is passed as arguments to `dsmc expire` commands.

Variables that change mmbackup output progress reporting

MMBACKUP_PROGRESS_CONTENT
Controls what progress information is displayed to the user as **mmbackup** runs. It is a bit field with the following bit meanings:

0x01
Specifies that basic text progress for each server is to be displayed.

0x02
Specifies that additional text progress for phases within each server is to be displayed.

0x04
Specifies that numerical information about files being considered is to be displayed.

MMBACKUP_PROGRESS_INTERVAL
Controls how frequently status callouts are made. The value is the minimum number of seconds between calls to the `MMBACKUP_PROGRESS_CALLOUT` script or program. It does not affect how frequently messages are displayed, except for the messages of `MMBACKUP_PROGRESS_CONTENT` category 0x04.

MMBACKUP_PROGRESS_CALLOUT
Specifies the path to a program or script to be called with a formatted argument, as described in the topic *MMBACKUP_PROGRESS_CALLOUT environment variable* in the *IBM Spectrum Scale: Administration Guide*.

Variables that change mmbackup debugging facilities
In case of a failure, certain debugging and data collection can be enabled by setting the specified environment variable value.

DEBUGmmbackup
This variable controls what debugging features are enabled. It is interpreted as a bit mask with the following bit meanings:

0x001
Specifies that basic debug messages are printed to STDOUT. Because **mmbackup** comprises multiple components, the debug message prefixes can vary. Some examples include:

```
mmbackup:mbackup.sh
DEBUGtsbackup33:
```

0x002
Specifies that temporary files are to be preserved for later analysis.

0x004
Specifies that all dsmc command output is to be mirrored to STDOUT.

DEBUGmmcmi
This variable controls debugging facilities in the **mmbackup** helper program **mmcmi**, which is used when the cluster minReleaseLevel is less than 3.5.0.11.

DEBUGtsbuhelper
This variable controls debugging facilities in the **mmbackup** helper program tsbuhelper, which is used when the cluster minReleaseLevel is greater than or equal to 3.5.0.11.

Variables that change mmbackup record locations

MMBACKUP_RECORD_ROOT
Specifies an alternative directory name for storing all temporary and permanent records for the backup. The directory name that is specified must be an existing directory and it cannot contain special characters (for example, a colon, semicolon, blank, tab, or comma).

The directory that is specified for MMBACKUP_RECORD_ROOT must be accessible from each node that is specified with the -N option.

Variable that defines the IBM Spectrum Protect server to be used for the backup process

MMBACKUP_SERVER_FROM_OPT
Specifies where to get the IBM Spectrum Protect server to be used when running the **mmbackup** command without the **--tsm-servers** option. Possible values are 0 and 1. Default value is 0.

If the value set for the **MMBACKUP_SERVER_FROM_OPT** variable is 1, the **mmbackup** command uses the **dsmc q opt SERVERNAME** command to get the IBM Spectrum Protect server that needs to be used, unless **--tsm-servers** option is specified. If **mmbackup --tsm-servers** option is specified, the **--tsm-servers** value will be used for backup, even if the value of the **MMBACKUP_SERVER_FROM_OPT** variable is set to 1.

If neither **MMBACKUP_SERVER_FROM_OPT** nor **--tsm-servers** is specified, the values defined in the dsm.sys file are used to determine the IBM Spectrum Protect server to be used for backup.

Exit status

0
Successful completion. All of the eligible files were backed up.

1
Partially successful completion. Some files, but not all eligible files, were backed up. The shadow database or databases reflect the correct inventory of the IBM Spectrum Protect server. Invoke **mmbackup** again to complete the backup of eligible files.

2
A failure occurred that prevented backing up some or all files or recording any progress in the shadow database or databases. Correct any known problems and invoke **mmbackup** again to complete the backup of eligible files. If some files were backed up, using the -q or --rebuild option can help avoid backing up some files additional times.

Security

You must have root authority to run the **mmbackup** command.

The node on which the command is issued, and all other IBM Spectrum Protect Backup-Archive client nodes, must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To perform an incremental backup of the file system gpfs0, issue the following command:

```
# mmbackup gpfs0
```

A sample output is as follows:

```
-----------------------------------------------------
mmbackup: Backup of /gpfs/gpfs0 begins at Mon Apr 7 15:37:50 EDT 2014.
-----------------------------------------------------
Mon Apr  7 15:38:04 2014 mmbackup:Scanning file system gpfs0
Mon Apr  7 15:38:14 2014 mmbackup:Determining file system changes for gpfs0 [balok1].
Mon Apr  7 15:38:14 2014 mmbackup:changed=364, expired=0, unsupported=0 for server [balok1]
Mon Apr  7 15:38:14 2014 mmbackup:Sending files to the TSM server [364 changed, 0 expired].
mmbackup: TSM Summary Information:
        Total number of objects inspected:      364
        Total number of objects backed up:      364
        Total number of objects updated:        0
        Total number of objects rebound:        0
        Total number of objects deleted:        0
        Total number of objects expired:        0
        Total number of objects failed:         0
        Total number of bytes transferred:      2179695902
-----------------------------------------------------
mmbackup: Backup of /gpfs/gpfs0 completed successfully at Mon Apr 7 15:41:09 EDT 2014.
-----------------------------------------------------
```

2. To re-create a shadow database for gpfs0, issue the following command:

```
# mmbackup gpfs0 --rebuild
```

A sample output is as follows:

```
-----------------------------------------------------
mmbackup: Shadow database rebuild of /gpfs/gpfs0 begins at Tue Apr 8 09:44:59 EDT 2014.
-----------------------------------------------------
Tue Apr  8 09:45:11 2014 mmbackup:Querying files currently backed up in TSM server:balok1.
Tue Apr  8 09:45:14 2014 mmbackup:Built query data file from TSM server: balok1 rc = 0
Tue Apr  8 09:45:17 2014 mmbackup:Scanning file system gpfs0
Tue Apr  8 09:45:26 2014 mmbackup:Reconstructing previous shadow file /gpfs/
gpfs0/.mmbackupShadow.1.balok1 from
 query data for balok1
Tue Apr  8 09:45:26 2014 mmbackup:Done with shadow file database rebuilds
-----------------------------------------------------
mmbackup: Shadow database rebuild of /gpfs/gpfs0 completed successfully at Tue Apr 8 09:45:26 EDT 2014.
-----------------------------------------------------
```

3. To perform an incremental backup of the file system gpfs0 with more progress information displayed, first issue the following command:

```
# export MMBACKUP_PROGRESS_CONTENT=3
```

Next, issue the **mmbackup** command:

```
# mmbackup gpfs0 --incremental-backup-threads 2 --selective-backup-threads 10
```

A sample output is as follows:

```
-----------------------------------------------------
mmbackup: Backup of /gpfs/gpfs0 begins at Fri Oct 26 13:12:18 EDT 2018.
-----------------------------------------------------
Fri Oct 26 13:12:21 2018 mmbackup:Begin checking server and shadow file for SERVER1
Fri Oct 26 13:12:52 2018 mmbackup:Querying TSM server SERVER1 for options
Fri Oct 26 13:12:53 2018 mmbackup:Found old shadow DB for SERVER1 present in /gpfs/gpfs0/.mmbackupShadow.1.SERVER1.filesys
Fri Oct 26 13:12:53 2018 mmbackup:Checking format version of old shadow DB
Fri Oct 26 13:12:53 2018 mmbackup:Found old shadow DB version: 1400
Fri Oct 26 13:12:53 2018 mmbackup:Previous shadow /gpfs/gpfs0/.mmbackupShadow.1.SERVER1.filesys state: present
Fri Oct 26 13:12:53 2018 mmbackup:Generating policy rules file:/var/mmfs/mmbackup/.mmbackupRules.gpfs0 to use /gpfs/gpfs0/.mmbackupCfg/
BAexecScript.gpfs0
Fri Oct 26 13:12:54 2018 mmbackup:Completed policy rule generation.  1 Exclude Dir directives, 0 Exclude File directives, 0 Include directives, 0
Warnings.
Fri Oct 26 13:12:54 2018 mmbackup:Scanning file system gpfs0
Fri Oct 26 13:13:04 2018 mmbackup:File system scan of gpfs0 is complete.
Fri Oct 26 13:13:04 2018 mmbackup:Calculating backup and expire lists for server SERVER1
Fri Oct 26 13:13:04 2018 mmbackup:Determining file system changes for gpfs0 [SERVER1].
Fri Oct 26 13:13:04 2018 mmbackup:changed=11, expired=10, unsupported=0, statechanged=20 for server [SERVER1]
Fri Oct 26 13:13:04 2018 mmbackup:Finished calculating lists [11 changed, 10 expired, 20 updated] for server SERVER1.
Fri Oct 26 13:13:04 2018 mmbackup:Sending files to the TSM server [11 changed, 10 expired, 20 updated].
Fri Oct 26 13:13:04 2018 mmbackup:Performing expire operations
Fri Oct 26 13:13:09 2018 mmbackup:Expiration job finished: processed:10 failed:0 rc:0
Fri Oct 26 13:13:09 2018 mmbackup:Completed policy expiry run with 0 policy errors, 0 files failed, 0 severe errors, returning rc=0.

Fri Oct 26 13:13:09 2018 mmbackup:Policy for expiry returned 0 Highest TSM error 0
Fri Oct 26 13:13:09 2018 mmbackup:Performing backup operations
Fri Oct 26 13:13:15 2018 mmbackup:Backup job finished: processed:11 failed:0 rc:0
Fri Oct 26 13:13:15 2018 mmbackup:Completed policy backup run with 0 policy errors, 0 files failed, 0 severe errors, returning rc=0.
```

```
Fri Oct 26 13:13:15 2018 mmbackup:Policy for backup returned 0 Highest TSM error 0
Fri Oct 26 13:13:15 2018 mmbackup:Performing backup operations
Fri Oct 26 13:13:21 2018 mmbackup:Backup job finished: processed:10 failed:0 rc:0
Fri Oct 26 13:13:21 2018 mmbackup:Backup job finished: processed:10 failed:0 rc:0
Fri Oct 26 13:13:21 2018 mmbackup:Completed policy backup run with 0 policy errors, 0 files failed, 0 severe errors, returning rc=0.
Fri Oct 26 13:13:21 2018 mmbackup:Policy for backup returned 0 Highest TSM error 0
mmbackup: TSM Summary Information:
        Total number of objects inspected:      41
        Total number of objects backed up:      31
        Total number of objects updated:        0
        Total number of objects rebound:        0
        Total number of objects deleted:        0
        Total number of objects expired:        10
        Total number of objects failed:         0
        Total number of objects encrypted:      0
        Total number of bytes inspected:        731973
        Total number of bytes transferred:      739603
Fri Oct 26 13:13:21 2018 mmbackup:analyzing: results from SERVER1.
Fri Oct 26 13:13:21 2018 mmbackup:Copying updated shadow file to the TSM server
Fri Oct 26 13:13:23 2018 mmbackup:Done working with files for TSM Server: SERVER1.
Fri Oct 26 13:13:23 2018 mmbackup:Completed backup and expire jobs.
Fri Oct 26 13:13:23 2018 mmbackup:TSM server SERVER1
        had 0 failures or excluded paths and returned 0.
        Its shadow database has been updated. Shadow DB state:updated
Fri Oct 26 13:13:23 2018 mmbackup:Completed successfully. TSM exit status:  exit 0

------------------------------------------------------------
mmbackup: Backup of /gpfs/gpfs0 completed successfully at Fri Oct 26 13:13:23 EDT 2018.
------------------------------------------------------------
```

4. To perform an incremental backup of the objects in the inode space of the /gpfs/testfs/infs2 directory to the balok1 server, issue the following command:

```
# mmbackup /gpfs/testfs/infs2  -t incremental --scope inodespace  --tsm-servers balok1
```

A sample output is as follows:

```
------------------------------------------------------------
mmbackup: Backup of /gpfs/testfs/infs2 begins at Wed May 27 12:58:39 EDT 2015.
------------------------------------------------------------
Wed May 27 12:58:48 2015 mmbackup:Scanning fileset testfs.indfs2
Wed May 27 12:58:53 2015 mmbackup:Determining fileset changes for testfs.indfs2 [balok1].
Wed May 27 12:58:53 2015 mmbackup:changed=2, expired=2, unsupported=0 for server [balok1]
Wed May 27 12:58:53 2015 mmbackup:Sending files to the TSM server [2 changed, 2 expired].
mmbackup: TSM Summary Information:
        Total number of objects inspected:      4
        Total number of objects backed up:      2
        Total number of objects updated:        0
        Total number of objects rebound:        0
        Total number of objects deleted:        0
        Total number of objects expired:        2
        Total number of objects failed:         0
        Total number of objects encrypted:      0
        Total number of bytes inspected:        53934
        Total number of bytes transferred:      53995
------------------------------------------------------------
mmbackup: Backup of /gpfs/testfs/infs2 completed successfully at Wed May 27 12:59:31 EDT
2015.
------------------------------------------------------------
```

5. To perform an incremental backup of a global snapshot that is called backupsnap6 to the balok1 server, issue the following command:

```
# mmbackup testfs -t incremental -S backupsnap6 --scope filesystem  --tsm-servers balok1
```

A sample output is as follows:

```
------------------------------------------------------------
mmbackup: Backup of /gpfs/testfs begins at Wed May 27 13:08:45 EDT 2015.
------------------------------------------------------------
Wed May 27 13:08:50 2015 mmbackup:Scanning file system testfs
Wed May 27 13:08:53 2015 mmbackup:Determining file system changes for testfs [balok1].
Wed May 27 13:08:53 2015 mmbackup:changed=130, expired=100, unsupported=0 for server [balok1]
Wed May 27 13:08:53 2015 mmbackup:Sending files to the TSM server [130 changed, 100 expired].

Wed May 27 13:08:59 2015 mmbackup:Policy for expiry returned 9 Highest TSM error 0
mmbackup: TSM Summary Information:
        Total number of objects inspected:      230
        Total number of objects backed up:      130
        Total number of objects updated:        0
        Total number of objects rebound:        0
        Total number of objects deleted:        0
        Total number of objects expired:        100
        Total number of objects failed:         0
        Total number of objects encrypted:      0
        Total number of bytes inspected:        151552
        Total number of bytes transferred:      135290
------------------------------------------------------------
```

```
mmbackup: Backup of /gpfs/testfs completed successfully at Wed May 27 13:09:05 EDT 2015.
-----------------------------------------------------------
```

See also

- “mmapplypolicy command” on page 80

Location

/usr/lpp/mmfs/bin

mmbackupconfig command

Collects GPFS file system configuration information.

Synopsis

```
mmbackupconfig Device -o OutputFile
```

Availability

Available on all IBM Spectrum Scale editions. Available on AIX and Linux.

Description

The `mmbackupconfig` command, in conjunction with the `mmrestoreconfig` command, can be used to collect basic file system configuration information that can later be used to restore the file system. The configuration information backed up by this command includes block size, replication factors, number and size of disks, storage pool layout, filesets and junction points, policy rules, quota information, and a number of other file system attributes.

This command does not back up user data or individual file attributes.

For more information about the `mmimgbackup` and `mmimgrestore` commands, see the topic *Scale out Backup and Restore (SOBAR)* in the *IBM Spectrum Scale: Administration Guide*.

Parameters

Device
The device name of the file system to be backed up. File system names need not be fully-qualified. `fs0` is as acceptable as `/dev/fs0`.

This must be the first parameter.

-o *OutputFile*
The path name of a file to which the file system information is to be written. This file must be provided as input to the subsequent `mmrestoreconfig` command.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmbackupconfig` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

To backup file system **fsiam2** to output file **backup.config.fsiam2** issue:

```
mmbackupconfig fsiam2 -o backup.config.fsiam2
```

The system displays information similar to:

```
mmbackupconfig: Processing file system fsiam2 ...
mmbackupconfig: Command successfully completed
```

See also

- “mmimgbackup command” on page 458
- “mmimgrestore command” on page 462
- “mmrestoreconfig command” on page 671

Location

/usr/lpp/mmfs/bin

mmbuildgpl command

Manages prerequisite packages for Linux and builds the GPFS portability layer.

Synopsis

```
mmbuildgpl [--quiet] [--build-package] [-v]
```

Availability

Available on all IBM Spectrum Scale editions. Available and needed only on Linux.

Description

Use the `mmbuildgpl` command to manage and verify prerequisite packages for Linux and build the GPFS portability layer. If all packages are installed correctly, `mmbuildgpl` builds the GPFS portability layer. If any packages are missing, the package names are displayed. The missing packages can be installed manually.

Tip: You can configure a cluster to rebuild the GPL automatically whenever a new level of the Linux kernel is installed or whenever a new level of IBM Spectrum Scale is installed. For more information, see the description of the **autoBuildGPL** attribute in the topic *mmchconfig command* in the *IBM Spectrum Scale: Command and Programming Reference*.

Parameters

--quiet
Specifies that when there are any missing packages, the `mmbuildgpl` command installs the prerequisite packages automatically by using the default package manager.

--build-package
Builds an installable package (`gpfs.gplbin`) for the portability layer binaries after compilation is successful. This option builds an RPM package on SLES and RHEL Linux and a Debian package on Debian and Ubuntu Linux.

When the command finishes, it displays the location of the generated package as in the following examples:

```
Wrote: /root/rpmbuild/RPMS/x86_64/gpfs.gplbin-3.10.0-229.el7.x86_64-5.1.1-x.x86_64.rpm
```

or

```
Wrote: /tmp/deb/gpfs.gplbin-4.4.0-127-generic_5.1.1-x_amd64.deb
```

You can then copy the generated package to other machines for deployment. By default, the generated package can be deployed only to machines whose architecture, distribution level, Linux kernel, and IBM Spectrum Scale maintenance level are identical with those of the machine on which the `gpfs.gplbin` package was built. However, you can install the generated package on a machine with a different Linux kernel by setting the `MM_INSTALL_ONLY` environment variable before you install the generated package. If you install the `gpfs.gplbin` package, you do not need to install the `gpfs.gpl` package.

Note: During the package generation, temporary files are written to the **/tmp/rpm** or **/tmp/deb** directory, so be sure there is sufficient space available. By default, the generated package goes to **/usr/src/packages/RPMS/<arch>** for SUSE Linux Enterprise Server, **/usr/src/redhat/RPMS/<arch>** for Red Hat Enterprise Linux, and **/tmp/deb** for Ubuntu Linux.

Important:

- The GPFS portability layer is specific to both the current kernel and the GPFS version. If either the kernel or the GPFS version changes, a new GPFS portability layer needs to be built.
- Although operating system kernels might upgrade to a new version, they are not active until after a reboot. Thus, a GPFS portability layer for this new kernel must be built after a reboot of the operating system.
- Before you install a new GPFS portability layer, make sure to uninstall the prior version of the GPFS portability layer first.

-v
Specifies that the output is verbose and contains information for debugging purposes.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmbuildgpl` command.

Examples

To build the GPFS portability layer, issue the **mmbuildgpl** command with no parameters, as in the following example:

```
# mmbuildgpl
-----------------------------------------------------
mmbuildgpl: Building GPL (5.0.3.0) module begins at Tue Mar 19 09:09:21 EDT 2019.
-----------------------------------------------------
Verifying Kernel Header...
kernel version = 31000229 (31000229000000, 3.10.0-229.el7.x86_64, 3.10.0-229)
module include dir = /lib/modules/3.10.0-229.el7.x86_64/build/include
module build dir = /lib/modules/3.10.0-229.el7.x86_64/build
kernel source dir = /usr/src/linux-3.10.0-229.el7.x86_64/include
Found valid kernel header file under /usr/src/kernels/3.10.0-229.el7.x86_64/include
Verifying Compiler...
make is present at /bin/make
cpp is present at /bin/cpp
gcc is present at /bin/gcc
g++ is present at /bin/g++
ld is present at /bin/ld
Verifying Additional System Headers...
Verifying kernel-headers is installed ...
Command: /bin/rpm -q kernel-headers
The required package kernel-headers is installed
make World ...
make InstallImages ...
-----------------------------------------------------
mmbuildgpl: Building GPL module completed successfully at Tue Mar 19 09:09:34 EDT 2019.
-----------------------------------------------------
```

See also

- *Building the GPFS portability layer on Linux nodes* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide.*

Location

`/usr/lpp/mmfs/bin`

mmcachectl command

Displays information about files and directories in the local page pool cache.

Synopsis

```
mmcachectl
    show [--device Device [{--fileset Fileset | --inode-num InodeNum [--show-lroc]}]] [--show-
filename] [-Y]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The **mmcachectl** command displays information about files and directories in the local page pool cache. You can display information for a single file, for the files in a fileset, or for all the files in a file system. If you issue `mmcachectl show` with no parameters, it displays information for all the files in all the file systems that have file data cached in the local page pool.

The command lists the following information for each file:

- The file system name.
- The fileset ID.
- The inode number.
- The snapshot ID.
- The file type (inode type): file, directory, link, or special.
- The number of instances of the file that are open.
- The number of instances of the file that are open for direct I/O.
- The file size in bytes.
- The size of the file data that is in the page pool in bytes.
- The size of the file data that is stored in LROC in bytes. Only available when `--inode-num` is specified.
- The cache location of the file attributes:

  F
  The file attributes are stored in the file cache.

  FD
  The file attributes are stored in the file cache and the file data is stored in the inode (data-in-inode). For information about data-in-inode, see the topic *Use of disk storage and file structure within a GPFS file system* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

  C
  The file attributes are stored in the stat cache. For information about the stat cache, see the topic *Non-pinned memory* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.
- The full path name of the file. For a file with multiple hard links, the command displays an asterisk (*) before the file name to indicate that it is one of the path names of the file. For more information, see the description of the `--show-filename` parameter later in this topic.

Parameters

`--device` *Device*
Specifies the device name of a file system for which you want to display information.

`--fileset` *Fileset*
Specifies the name of a fileset for which you want to display information.

--inode-num *InodeNum*
Specifies the inode number of a file for which you want to display information.

--show-lroc
Shows the data stored in the LROC device for the file whose inode number is given.

--show-filename
Causes the command to display the full path name of each file.

Note:

- This parameter is supported only on Linux nodes with a Linux kernel of 3.10.0-0123 or later and is dependent on the path name being available in the Linux directory cache.
- The parameter is not supported on AIX or on the Windows operating system.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmcachectl` command.

The node on which you enter the command must be able to execute remote shell commands on any other administration node in the cluster. It must be able to do so without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

Examples

1. The following command lists information for all the files in all the file systems that have file data cached in the local page pool. In this example, the only file system that has data cached in the local page pool is `gpfs_fs0`:

```
# mmcachectl show
FSname      Fileset  Inode    SnapID   FileType     NumOpen    NumDirect  Size         Cached
Cached
            ID                                     Instances  IO         (Total)      (InPagePool)
(InFileCache)
-------------------------------------------------------------------------------------------------------
--------
gpfs_fs0     0        40       0        special      1          0          524288       8192          F
gpfs_fs0     0        56833    0        file         0          0          0            0             F
gpfs_fs0     0        4        0        special      1          0          524288       524288        F
gpfs_fs0     0        18229    0        link         0          0          14           0             FD
gpfs_fs0     0        3        0        directory    0          0          262144       262144        F
gpfs_fs0     0        56836    0        directory    0          0          131072       0             F
gpfs_fs0     0        56832    0        file         0          0          166611       0             F
gpfs_fs0     0        39       0        special      1          0          524288       8192          F
gpfs_fs0     0        40704    0        file         0          0          10           0             FD
gpfs_fs0     0        41       0        special      1          0          524288       8192          F
gpfs_fs0     0        55554    0        directory    0          0          3968         0             FD
gpfs_fs0     0        42       0        special      0          0          524288       524288        F
gpfs_fs0     0        55552    0        file         0          0          5            0             FD
gpfs_fs0     0        56834    0        file         0          0          8192000000   0             F
gpfs_fs0     0        55553    0        file         0          0          13           0             FD
gpfs_fs0     0        22016    0        file         0          0          10           0             FD
```

2. The following command lists information for all the files in all the file systems that have file data cached in the local page pool. Because the **--show-filename** parameter is specified, the command also lists the full path name of each file in the last column:

```
# mmcachectl show --show-filename
FSname       Fileset  Inode    SnapID   FileType     NumOpen    NumDirect  Size         Cached         Cached          FileName
             ID                                      Instances  IO         (Total)      (InPagePool)   (InFileCache)
-------------------------------------------------------------------------------------------------------------------------------
gpfs_fs0     0        40       0        special      1          0          524288       8192           F             -
gpfs_fs0     0        56833    0        file         0          0          0            0              F             * /gpfs_fs0/big_rand_link1
gpfs_fs0     0        4        0        special      1          0          524288       524288         F               -
gpfs_fs0     0        18229    0        link         0          0          14           0              FD              /gpfs_fs0/test2_soft_link
gpfs_fs0     0        3        0        directory    0          0          262144       262144         F               /gpfs_fs0/
gpfs_fs0     0        56836    0        directory    0          0          131072       0              F               /gpfs_fs0/temp_dir
gpfs_fs0     0        56832    0        file         0          0          166611       0              F               /gpfs_fs0/bis_csc
gpfs_fs0     0        39       0        special      1          0          524288       8192           F               -
gpfs_fs0     0        40704    0        file         0          0          10           0              FD              /gpfs_fs0/test
gpfs_fs0     0        41       0        special      1          0          524288       8192           F               -
gpfs_fs0     0        55554    0        directory    0          0          3968         0              FD              /gpfs_fs0/dir
gpfs_fs0     0        42       0        special      0          0          524288       524288         F               -
gpfs_fs0     0        55552    0        file         0          0          5            0              FD              /gpfs_fs0/test1
gpfs_fs0     0        56834    0        file         0          0          8192000000   0              F               /gpfs_fs0/big_zero
gpfs_fs0     0        55553    0        file         0          0          13           0              FD              /gpfs_fs0/test2
gpfs_fs0     0        22016    0        file         0          0          10           0              FD              /gpfs_fs0/test3
```

3. The following command lists information for a specific file whose **--inode-num** is mentioned.

```
# mmcachectl show --device fs0 --inode-num 263174 --show-lroc
FSname       Fileset  Inode    SnapID   FileType     NumOpen    NumDirect  Size
Cached         Cached           Cached
             ID                                      Instances  IO         (Total)
(InPagePool)   (InFileCache)   (InLroc)
-------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------
fs0          0        263174   0        file         0          0          2041076846
348127232      F               1692949614

File count: 1
```

See also

- *mmfsadm command* in the *IBM Spectrum Scale: Problem Determination Guide*.

Location

/usr/lpp/mmfs/bin

mmcallhome command

Manages the call home operations.

Synopsis

```
mmcallhome group add groupName server  [--node {all | childNode[,childNode...]}]
```

or

```
mmcallhome group list [--long] [-Y]
```

or

```
mmcallhome group delete GroupName
```

or

```
mmcallhome group auto [--server ServerName1[,ServerName2...]]
                      [--nodes {all | ChildNode1[,ChildNode2...]}]
                      [--force]
                      [--group-names  {group1[,group2...]}]
                      [--enable [ {LICENSE | ACCEPT}] |--disable]
```

or

```
mmcallhome group change  GroupName {--add-nodes childNode1[,childNode2...]|
                                    --delete-nodes childNode1[,childNode2...]}...
```

or

```
mmcallhome capability list [-Y]
```

or

```
mmcallhome capability enable [{LICENSE | ACCEPT}]
```

or

```
mmcallhome capability disable
```

or

```
mmcallhome info list [-Y]
```

or

```
mmcallhome info change  { --customer-name  CustomerName | --customer-id CustomerId
                        | --email Email | --country-code CountryCode
                        | --type { production | test } }...
```

or

```
mmcallhome proxy enable [--with-proxy-auth]
```

or

```
mmcallhome proxy disable
```

or

```
mmcallhome proxy list [-Y]
```

or

```
mmcallhome proxy change { --proxy-location  ProxyLocation | --proxy-port ProxyPort
                        | --proxy-username  ProxyUsername | --proxy-password  ProxyPassword
} ...
```

or

```
mmcallhome schedule list [-Y]
```

or

```
mmcallhome schedule add --task { DAILY | WEEKLY | EVENT }
```

or

```
mmcallhome schedule delete --task { DAILY | WEEKLY | EVENT }
```

or

```
mmcallhome run GatherSend --task { DAILY | WEEKLY }
```

or

```
mmcallhome run SendFile --file file [--desc DESC | --pmr {xxxxx.yyy.zzz | TSxxxxxxxxx}]
```

or

```
mmcallhome status list [--task { DAILY  | WEEKLY | SENDFILE  | SENDPMRDATA }]
                       [--numbers num] [--verbose ] [-Y]
```

or

```
mmcallhome status diff [--last-days num]|[--old dcFileName [--new dcFileName]]
                       [--verbose] [-Y]
```

or

```
mmcallhome status delete {--task { DAILY  | WEEKLY  | SENDFILE  | SENDPMRDATA  } |
                          --startTime time | --startTimeBefore time | --all }
```

or

```
mmcallhome test connection
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the mmcallhome command to configure, enable, run, schedule, and monitor call home related tasks in the IBM Spectrum Scale cluster.

By using this command, predefined data from each node can be collected on a regular basis or on demand and uploaded to IBM. IBM support and development teams can use this data to understand how the customers are using IBM Spectrum Scale. In case of issues, the data can be referenced for problem analysis. The data can also possibly be used to provide advice to customers regarding failure prevention.

Since an IBM Spectrum Scale cluster consists of multiple nodes, the call home feature introduces the concept of the call home group to manage them. A *call home group* consists of one gateway node (which is defined as a call home node) and one or more client nodes (which are defined as call home child nodes). The call home node initiates the data collection from the call home child nodes and uploads data to IBM using the HTTPS protocol. Since the call home group can be configured independently, the group concept can be used for special conditions, such as for split clusters, that require all the group members to be on the same side to avoid unnecessary data transfer over large distance. Also, a call home group can be mapped to a node group or other cluster-specific attributes. The call home node needs to have access to the external network via port 443. The maximum recommended number of nodes per group is 128. Each cluster node can be only a member of at most one group. Multiple call home groups can be defined within an IBM Spectrum Scale cluster.

For more information about the call home feature, see the *Monitoring the IBM Spectrum Scale(tm) system remotely by using call home* section in *IBM Spectrum Scale: Problem Determination Guide*.

Parameters

group
Manages topology with one of the following actions:

add
Creates a call home group, which is a group of nodes consisting of one call home node and multiple call home child nodes. Multiple call home groups can be configured within an IBM Spectrum Scale cluster.

The call home node initiates data collection within the call home group and uploads the data package to the IBM server.

group
Specifies the name of the call home group.

Note: The group name can consist of any alphanumeric characters and these non-alphanumeric characters: '-', '_', and '.'.

Important: The group name cannot be `global`. Call home uses `global` as the default name for the group that contains the global values that are applied to all groups.

server
Specifies the name of the call home server belonging to the call home group.

Note: The server name can consist of any alphanumeric characters and these non-alphanumeric characters: '-', '_', and '.

--node childNode
Specifies the call home child nodes.

Note: The child node name can consist of any alphanumeric characters and these non-alphanumeric characters: '-', '_', and '.

--node all
Selects all Linux nodes in the IBM Spectrum Scale cluster. When this parameter is omitted, only the call home node is added to the child node. Additionally, call home node is always added to the child node group.

list
Displays the configured call home groups.

Note:

If nodes that are members of a call home group are deleted, or their long admin node names (including domain) are changed, the **`mmcallhome group list`** command displays `------` instead of the names of such nodes. In such cases, you must delete the corresponding groups, and then create new groups if needed. The deletion of the call home groups is not done automatically, since in some cases this might cause the deletion of the call home groups without re-creating them.

--long
Displays the node names as long admin node names. Without *--long* the node names are listed as short admin node names.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

delete
Deletes the specified call home group.

GroupName
Specifies the name of the call home group that needs to be deleted.

auto
Enables automatic creation of call home groups.

--server ServerName
Specifies one or more call home servers. Each server must be able to access the IBM call home servers over the internet. If no server is specified, the system detects call home node automatically. In this scenario, the system checks if the detected node can access the internet. If a server is specified, the defined nodes are used as call home nodes without any further check.

If a proxy is needed, specify the proxy by using the **mmcallhome proxy** command before running the **mmcallhome group auto** command.

Note: If this option is specified, the corresponding nodes are assumed to be able to access the IBM call home servers over the internet and no further checks in this regard is performed. If this option is not used, the **mmcallhome** command tests all potential call home nodes for the connectivity to the IBM call home servers. In this case, if a proxy is configured via the **mmcallhome proxy** command, this proxy is used to check the connectivity, otherwise a direct connection is attempted.

--nodes
Specifies the names of the call home child nodes to distribute into groups.

all
Specifies that all cluster nodes must be distributed into call home groups.

--force
Creates new groups after deleting the existing groups.

Note: If this option is selected but no server nodes are specified using the `--server` option or detected automatically, the operation is aborted and the existing groups are not deleted.

--group-names
Specifies the names of the call home groups to create. The number of call home group names must be bigger or equal to the number of created call home groups or the execution of the **mmcallhome group auto** command is aborted with an error. If this option is not specified, the automatically created groups are named in the following way: autoGroup_1, autoGroup_2, ...

--enable
Enables the cluster for call home functionality. If no other option is defined, the **enable** parameter shows the license and asks for acceptance by default.

LICENSE
Shows license and terminate.

ACCEPT
Does not show license and assumes that the license is accepted.

--disable
Disable call home.

Note: All groups are disabled.

change
Changes an existing call home group by adding new child nodes to it or removing existing child nodes from the group.

GroupName
Specifies the name of an existing call home group.

--add-nodes *ChildNode1*[,*ChildNode2*...]
Specifies the cluster nodes to add to the call home group as call home child nodes.

--delete-nodes *ChildNode1*[,*ChildNode2*...]
Specifies the call home child nodes that belong to the specified group, and must be removed.

capability
Manages the overall call home activities with one of the following actions:

list
Displays the configured customer information such as the current enable or disable status, call home node, and call home child nodes.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

enable
Enables the call home service.

Note: When you run the **mmcallhome capability enable** command, you are required to accept a data privacy disclaimer. For more information, see the *Data privacy with call home* section in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

disable
Disables the currently running call home service.

info
Manages customer data with one of the following actions:

list
Displays the configured parameter values.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

change
Sets parameter values.

[--key value]
Indicates a placeholder pointing to the following table:

Table 11. Key-value

Key	Value
customer-name	Business/company name This name can consist of any alphanumeric characters and these non-alphanumeric characters: '-', '_', '.', ' ', ','
customer-id	Customer ID This can consist of any alphanumeric characters and the following non-alphanumeric characters: • - • _ • . In special cases, the following customer IDs should be used: • For developer edition: DEVLIC • For test edition for customers who are trying IBM Spectrum Scale before buying: TRYBUY
email	Group or task e-mail address only. Ensure that the call home contact information is not a personal e-mail address, and is directed towards a group or task e-mail address. For example, itsupport@mycompanyname.com.
country-code	The country code in the ISO 3166-1 alpha-2 format. For example, US for USA, DE for Germany. For more details, see ISO 3166-1 alpha-2. The country code must correspond to the contact country. The contact country information can be found in the same source where the customer ID is found. Since the customer number is not unique throughout all countries, a country code is also required to unambiguously identify a customer.
type	Marks the respective call home cluster as a test or a production system. In case this parameter is not explicitly set, the value is set to *production* by default.

proxy

Configures proxy-related parameters with one of the following actions:

enable

Enables call home to use a proxy for its uploads. Requires the call home settings `proxy-location` and `proxy-port` to be defined.

[--with-proxy-auth]

Enables user ID and password authentication to the proxy server. Requires the call home settings proxy-username and proxy-password to be set.

disable

Disables proxy access.

list

Displays the currently configured proxy-related parameter values.

-Y

Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

change

Modifies the proxy configuration.

[--key value]

Indicates a placeholder pointing to the table below.

Table 12. key-value

Key	Value
proxy-location	Proxy server address (IP address/fully qualified domain name)
proxy-port	Proxy server port number
proxy-username	Proxy server user name This name can consist of any alphanumeric and non-alphanumeric characters.
proxy-password	Proxy server password This can consist of any alphanumeric and non-alphanumeric characters.

schedule

Configures scheduling of call home tasks with one of the following actions:

list

Shows if the scheduled data collection tasks or the event-based uploads feature are enabled. For more information about the collected information, see the *Types of call home data upload* section in the *IBM Spectrum Scale: Problem Determination Guide*

-Y

Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

add --task *{DAILY | WEEKLY | EVENT}*

Enables the specified scheduled task. The specified task can be a daily or weekly data collection and upload operation, or an event-based mini-snap creation.

delete --task *{DAILY | WEEKLY | EVENT}*

Disables the specified scheduled task. The specified task can be a daily or weekly data collection and upload operation, or an event-based mini-snap creation.

run

Executes one-time gather or send tasks with one of the following options:

GatherSend

Executes a one-time gather-send task, which collects the predefined data and uploads it.

--task daily
Chooses the data to be uploaded, as specified under the `daily` and `all` data collection schedules. For more information about the `daily` and `all` data collection schedules, see the *Scheduled data upload* section in the *IBM Spectrum Scale: Problem Determination Guide*.

--task weekly
Chooses the data to be uploaded, as specified under the `weekly` and `all` data collection schedules. For more information about the `daily` and `all` data collection schedules, see the *Scheduled data upload* section in the *IBM Spectrum Scale: Problem Determination Guide*.

SendFile
Uploads a specified file to IBM. The maximum file size is limited to 8 GiB from the IBM server side, but this limit might be increased in the future.

--file file
Specifies the name of the file that needs to be uploaded.

Note: The name can consist of any alphanumeric characters and these non-alphanumeric characters: '-', '_', '.'

[--desc DESC]
Specifies the description of the file that needs to be uploaded. This is added to the data package file name.

Note: This text can consist of any alphanumeric characters and these non-alphanumeric characters: '-', '_', '.', ' ', ','

[--pmr {xxxxx.yyy.zzz | TSxxxxxxxxx}]
Specifies either the dot-delimited PMR descriptor, where x, y and z could be digits, and y might additionally be a letter, or a Salesforce case descriptor, where each x is a digit.

status
Displays status of the call home tasks with one of the following options:

list
Displays the status of the currently running and the already completed call home tasks.

--task {DAILY | WEEKLY | SENDFILE | SENDPMRDATA}
Specifies the requested call home task type. The following types are available:

DAILY
Daily executed scheduled uploads.

WEEKLY
Weekly executed scheduled uploads.

SENDFILE
Files that are sent on demand, that are not PMR related or Salesforce case related.

SENDPMRDATA
Files that are sent on demand, that are either PMR related or Salesforce case related.

[--numbers num]
Specifies the maximum number of latest tasks that can be listed for each requested task type.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

--verbose
Lists additional information.

When the **mmcallhome status list --verbose** command is executed, the following information is shown in the output:

- **Group**: The name of the call home group where the task was executed.
- **Task**: Type of the executed call home task: DAILY, WEEKLY, SENDFILE or SENDPMRDATA.
- **Start Time**: A time stamp, specifying the start time of the call home task. This timestamp is also used to uniquely identify a call home task within a call home group.
- **Updated Time**: A time stamp of the last status update for the corresponding task.
- **Status**: You can get one of the following values as status information:
  - `success`
  - `running`
  - `failed`
  - `minor error`
  - `aborted`
- **RC or Step**:
  - For a task that is currently running, the current execution step is displayed.
  - For a successful task, `RC=0` is displayed.
  - For a failed task, the failure details are displayed.
- **Package File Name**: Name of the created data package to be uploaded.
- **Original Filename**: Name of the transferred file for the `SendFile` tasks.

diff

Displays the configuration difference between any of the following data collection files:

- *DAILY*
- *WEEKLY*
- *HEARTBEAT*

When this command is specified without any option other than --verbose or `-Y`, it compares the latest two call home data collection files, and prints out the difference.

--last-days *num*

Specifies the call home data collection file that was created *num* days back to compare with the most recent one.

--old *dcFileName*

Specifies the file name of the call home data collection file to compare with the most recent file. The first file that matches is selected. Either the full filename or just a substring such as the creation date in the format 20200229 can be specified.

Note: If the next option **`--new`** *<dcFileName>* is also specified, this data collection file is selected instead of the most recent data file to compare with the old call home data collection file.

--new *dcFileName*

This parameter can only be used in combination with **`--old`** *dcFileName* option. If denoted, the specified data collection file is compared with the file specified using the **`--old`** option, instead of the most recent one. The selected file must be newer than the one specified with **`--old`** option.

--verbose

Print out a more verbose output. Instead of just the name of the configuration object that was deleted or created all its detailed fields are printed out.

-Y

Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

delete
Deletes the status log entries for the specified tasks:

--task {DAILY | WEEKLY | SENDFILE | SENDPMRDATA}
Specifies the task type, for which the status log entries should be deleted.

--startTime starttime
Specifies the start time of the log to delete.

--startTimeBefore starttime
All logs older than the time specified by this option are deleted.

--all
All logs are deleted.

test
Executes detailed system checks:

connection

Checks the connectivity to the IBM e-support infrastructure. A proxy is used if the call home proxy settings are enabled. If the proxy setting is disabled, direct connections are attempted.

If this command is executed from a node that is a member of a currently existing call home group, the system performs the connectivity check for the call home master node of this group. If this command is executed from a node that is not a member of any existing call home group, the system performs a connectivity check for this node, and checks if it can become a call home master node.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmcallhome` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To configure a call home group, issue the following command:

```
# mmcallhome group add group1 test-11 --node test-11,test-12,test-13
```

A sample output is as follows:

```
Call home group group1 has been created
```

2. To view the configured call home groups, issue the following command:

```
# mmcallhome group list
```

A sample output is as follows:

```
Call Home Group   Call Home Node   Call Home Child Nodes
----------------- ---------------- ------------------------
group1            test-11          test-11,test-12,test-13
```

You can also give the same command with the `--long` option to view the configured call home groups with their long names:

```
# mmcallhome group list --long
```

A sample output is as follows:

```
Call Home Group   Call Home Node         Call Home Child
Nodes
----------------- ---------------------- ------------------------------------------
group1            test-11.localnet.com   test-11.localnet.com,test-12.localnet.com,
                                         test-13.localnet.com
```

3. To change customer information such as customer name, customer ID, country code, and the system type, issue the following command:

```
# mmcallhome info change --customer-name "SpectrumScaleTest"
--customer-id "1234" --country-code "JP" --type production
```

A sample output is as follows:

```
Success
Call home country-code has been set to JP
Call home customer-id has been set to 1234
Call home customer-name has been set to SpectrumScaleTest
Call home type has been set to production
```

4. To view the customer information, issue the following command:

```
# mmcallhome info list
```

A sample output is as follows:

```
group    customerName        customerID   email      countryCode   type
-------- ------------------- ------------ ---------- ------------- ------
global   SpectrumScaleTest   1234         unknown    JP            production
```

5. To create a call home group automatically, issue the following command:

```
# mmcallhome group auto
```

A sample output is as follows:

```
[I] Analysing the cluster...
[I] Creating <1> new call home groups:
[I] Call home child nodes = g5020-31.localnet.com,
    g5020-32.localnet.com,g5020-34.localnet.com
[I] Call home group autoGroup_1 has been created
[I] Nodes without call home: <1> (g5020-33.localnet.com)
[I] Updating call home node classes...
    g5020-32.localnet.com:
    QOS configuration has been installed and broadcast to all nodes.
    g5020-32.localnet.com:
    QOS configuration has been installed and broadcast to all nodes.
[I] The automatic group creation completed successfully.
```

6. To create a call home group automatically and enable the cluster for call home functionality by displaying options for acceptance, issue the following command:

```
# mmcallhome group auto --enable
```

A sample output is as follows:

```
By accepting this request, you agree to activate the Call Home Feature of the  Program and
allow IBM and its subsidiaries to store and use your contact information and your support
information anywhere they do business worldwide as further  described in the Program
```

```
license agreement and the documentation page "Data privacy with Call Home". If you agree,
please respond with "accept" for acceptance, else with "not accepted" to decline.

(accept / not accepted)
accept
[I] Analysing the cluster...
[I] Creating <1> new call home groups:
[I] Call home child nodes = g5020-31.localnet.com,
    g5020-32.localnet.com,g5020-34.localnet.com
[I] Call home group autoGroup_1 has been created
[I] Nodes without call home: <1> (g5020-33.localnet.com)
[I] Updating call home node classes...
g5020-32.localnet.com: QOS configuration has
been installed and broadcast to all nodes.
g5020-32.localnet.com: QOS configuration has
been installed and broadcast to all nodes.
[I] The automatic group creation completed successfully.
Call home enabled has been set to true

Additional messages:
License was accepted. Call home enabled.
```

Note: To accept the call home functionality, type **accept** manually. Type `mmcallhome group auto --enable accept` to avoid the explicit acceptance from the user.

7. To use create new group after deleting the existing group, issue the following command:

```
# mmcallhome group auto --force
```

A sample output is as follows:

```


[I] Analysing the cluster...
[I] Deleting old call home groups (--force mode)
[I] Creating <1> new call home groups:
[I] Call home child nodes = g5020-31.localnet.com,
    g5020-32.localnet.com,g5020-34.localnet.com
[I] Call home group autoGroup_1 has been created
[I] Nodes without call home: <1> (g5020-33.localnet.com)
[I] Updating call home node classes...
[I] The automatic group creation completed successfully.
```

8. To enable the call home service, issue the following command:

```
# mmcallhome capability enable
```

A sample output is as follows:

```


By accepting this request, you agree to activate the Call Home Feature of the  Program and
allow IBM and its subsidiaries to store and use your contact information and your support
information anywhere they do business worldwide as further  described in the Program
license agreement and the documentation page "Data privacy with Call Home". If you agree,
please respond with "accept" for acceptance, else with "not accepted" to decline.

(accept / not accepted)

accept
Call home enabled has been set to true

Additional messages:
License was accepted. Callhome enabled.
```

9. To register a daily task with cron, issue the following command:

```
# mmcallhome schedule add --task daily
```

A sample output is as follows:

```
Call home daily has been set to enabled
```

10. To register a weekly task with cron, issue the following command:

```
# mmcallhome schedule add --task weekly
```

A sample output is as follows:

```
Call home weekly has been set to enabled
```

11. To list the registered tasks for gather-send, issue the following command:

```
# mmcallhome schedule list
```

A sample output is as follows:

```
group    scheduleType   isEnabled
-------- -------------- -----------
global   daily          enabled
global   weekly         enabled
global   event          enabled
```

12. To set the parameters for the proxy server, issue the following command:

```
# mmcallhome proxy change --proxy-location okapi --proxy-port 80
--proxy-username root --proxy-password <password>
```

A sample output is as follows:

```
Call home proxy-port has been set to 80
Call home proxy-username has been set to root
Call home proxy-password has been set to <password>
Call home proxy-location has been set to okapi
```

13. To view the proxy server parameters, issue the following command:

```
# mmcallhome proxy list
```

A sample output is as follows:

```
group     proxyAuthEnabled  proxyEnabled   proxyLocation   proxyPort  proxyUsername
-------- ------------------ ------------- --------------  ----------- --------------
global         false             false         okapi           80          root
```

14. To invoke a one-time gather-send task, issue the following command:

```
# mmcallhome run GatherSend --task daily
```

A sample output is as follows:

```
One time run completed with success
```

15. To run one-time send command to upload a file, issue the following command:

```
# mmcallhome run SendFile --file /ibm/gpfs0/testDir/testFile --desc MyTestData
```

A sample output is as follows:

```
Running sendFile... (In case of network errors, it may take over 20 minutes for retries.)
Successfully uploaded the given file
Run mmcallhome status list --verbose to see the package name
```

16. To view the status of the currently running and the already completed call home tasks, issue the following command:

```
# mmcallhome status list
```

A sample output is as follows:

```
=== Executed call home tasks ===
```

```
Group         Task         Start Time           Status
-------------------------------------------------------
autoGroup_1   daily        20181212021402.015   success
autoGroup_1   weekly       20181203110048.724   success
autoGroup_1   sendfile     20181203105920.936   success
autoGroup_1   SendPMRData  20181212141347.236   success
```

17. To test the connection, issue the following command:

```
# mmcallhome test connection
```

A sample output is as follows:

```
Starting connectivity test between the call home node and IBM
Call home node: g5020-31.localnet.com
Starting time: Fri Aug 31 17:09:58 2018

Testing direct connection
------------------------------------------------------------
Testing <prefix Edge_SP_Config>:
Edge_SP_Config_1: 129.42.56.189 OK

Testing <prefix Edge_Profile>:
Edge_Profile_1: 129.42.56.189 OK

Testing <prefix Edge_Status_Report>:
Edge_Status_Report_1: 129.42.56.189 OK

------------------------------------------------------------
End time: Fri Aug 31 17:10:06 2018
```

18. To list the differences between two call home data collection files of different date, issue the following command:

```
# mmcallhome status diff --new 20200131 --old 20200129 -v
```

A sample output is as follows:

```
Cluster Data (modified)
Sdrfs Gennumber  : 43  --> 55


Fs Data

Device Name  =  gpfs0 (modified)
Quota Option         : no                --> userquota;groupquota;filesetquota;perfileset
Perfileset Quota     : no                --> yes
Mount Options        : rw,relatime,mtime  -->
rw,relatime,mtime,userquota;groupquota;filesetquota;perfileset

Device Name  =    mari (created)
Rw Options           : rw
Mount Options        : rw,atime,mtime,userquota;groupquota;filesetquota,nfssync,nodev
Drive Letter         : N/A
Datatype             : sdrq_fs_info
Perfileset Quota     : no
Mount Point          : /mnt/mari
Device Name          : mari
Quota Option         : userquota;groupquota;filesetquota
Automount Option     : yes
Mtime Option         : mtime
Fs Type              : local
Dmapi Enabled        : no
Atime Option         : atime
Maintenance Mode     : no
Device Minor Number  : 152
Other Mount Options  : nfssync,nodev
Owning Cluster Name  : gpfs-cluster-2.localnet.com
Remote Device Name   : mari


Nodeclass Data

Nodeclass  =  GUI_MGMT_SERVERS (modified)
Allmembers   : mari-23.localnet.com  --> mari-23.localnet.com,mari-22.localnet.com
Membernodes  : mari-23.localnet.com  --> mari-23.localnet.com,mari-22.localnet.com
```

See also

- “mmchconfig command” on page 170
- “mmlscluster command” on page 492
- “mmlsconfig command” on page 495
- “mmnfs command” on page 560
- “mmobj command” on page 573
- “mmsmb command” on page 709
- “mmuserauth command” on page 738

Location

`/usr/lpp/mmfs/bin`

mmces command

Manage CES (Cluster Export Services) configuration.

Synopsis

```
mmces service enable {NFS | OBJ | SMB | BLOCK | HDFS}
```

or

```
mmces service disable {NFS | OBJ | SMB | BLOCK | HDFS}[--force]
```

or

```
mmces service {start | stop} {NFS | OBJ | SMB | BLOCK | HDFS}
              [-N {Node[,Node...] | NodeFile | NodeClass} | -a]
```

or

```
mmces service list [-N {Node[,Node...] | NodeFile | NodeClass} | -a] [-Y] [--verbose]
```

or

```
mmces node {suspend [--stop ] | resume [--start ]}
           [-N {Node[,Node...] | NodeFile | NodeClass} | -a]
```

or

```
mmces node list [-Y] [--ces-group Group[,Group...]] [--verbose]
```

or

```
mmces log level [new-level]
```

or

```
mmces address add [--ces-node Node] [--attribute Attribue]
                  [--ces-group Group] --ces-ip {IP[,IP...]}
```

or

```
mmces address remove --ces-ip {IP[,IP...]}
```

or

```
mmces address move --ces-ip {IP[,IP...]} --ces-node Node
```

or

```
mmces address move --rebalance
```

or

```
mmces address change --ces-ip IP [,IP...]
                     [--attribute Attribute[,Attribute...]]
                     [--ces-group Group]
```

```
mmces address change --ces-ip IP [,IP...]
                     [--remove-attribute]
                     [--remove-group]
```

or

```
mmces address list [-N {Node[,Node...]| NodeFile | NodeClass} | -a] [-Y]
```

```
mmces address list [--ces-ip IP[,IP...]] [--ces-group Group[,Group...]]
                   [--attribute Attribute[,Attribute...]]
                   [--by-node | [--extended-list | --full-list] -Y]
```

or

```
mmces address policy [none | balanced-load | node-affinity | even-coverage]
```

or

```
mmces address prefix-ipv4 { prefix-value | show }
```

or

```
mmces state show {NFS | OBJ | SMB | BLOCK | AUTH | AUTH_OBJ
                  |NETWORK | CES | HDFS | HDFS_NAMENODE}
                  [-N {Node[,Node...] | NodeFile | NodeClass} | -a] [-Y]
```

or

```
mmces state cluster {NFS | OBJ | SMB | BLOCK | AUTH | AUTH_OBJ
                     |NETWORK | CES | HDFS | HDFS_NAMENODE} [-Y]
```

or

```
mmces interface mode {interface |legacy}[--force]
```

or

```
mmces interface mode legacy --clear[--force]
```

or

```
mmces interface list [--by-node Node1[,Node2...]]
                     [--by-group {Group1[,Group2...]| ALL | ANY | NONE }]
```

or

```
mmces interface add --nic NIC1, [NIC2...]
                    [--ces-group Group1[,Group2...] ]
                    [--ces-node Node1[,Node2...]]
                    [--force]
```

or

```
mmces interface remove --nic NIC1[,NIC2...]
                    [--ces-node Node1[,Node2...]]
                    [--force]
```

or

```
mmces interface check
```

or

```
mmces events active [NFS | OBJ | SMB | BLOCK | AUTH | AUTH_OBJ |NETWORK | HDFS]
                     [-N {Node[,Node...] | NodeFile | NodeClass} | -a]
```

or

```
mmces events list [NFS | OBJ | SMB | BLOCK | AUTH | AUTH_OBJ |NETWORK | HDFS]
                  [-Y] [--time {hour | day | week | month}]
                  [--severity {INFO | WARNING | ERROR | SEVERE}]
                  [-N {Node[,Node...] | NodeFile | NodeClass} | -a]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmces` command to manage protocol addresses, services, node state, logging level and load balancing. Assignment of the CES addresses to the network adapters is done by the CES base address specified in `CES legacy mode`, or by a user-specified configuration in the `CES interface mode`. The protocol functions provided in this command, or any similar command, are generally referred to as CES (Cluster Export Services). For example, protocol node and CES node are functionally equivalent terms. Before using the **mmces** command, CES has to be installed and configured in the system. For more information on how to install CES, see the *Installing Cluster Export Services as part of installing IBM Spectrum Scale on Linux systems* section in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

CES currently supports the NFS, SMB, BLOCK, HDFS, and Object services. Each service can be enabled or disabled with the **mmces service** command. Enabling a service is a CES cluster-wide operation. In addition, enabled services can be started and stopped on individual nodes.

Clients access the CES services using one or more IP addresses in the CES address pool. Addresses can be added to and removed from the pool using the **mmces address add** and **mmces address remove** commands. Existing addresses can be reassigned to another node with the **mmces address move** command.

Addresses can have one or more attributes associated with them. An address attribute is a tag that the services can identify a specific address as having a special meaning, which is defined by the service protocol. Addresses can have multiple attributes, but an attribute can only be associated with a single address. The supported attributes are `object_singleton_node`, `object_database_node`, and HDFS-related attributes. The HDFS-related attributes are assigned automatically. These attributes are used to manage HDFS and object protocol-related services. For more information, see the *Object services and object protocol nodes* table in the *Understanding and managing Object services* section in the *IBM Spectrum Scale: Administration Guide*.

Addresses can have a policy associated with them, and that policy determines how addresses are automatically distributed. The allowed policies are `none`, `balanced-load`, `node-affinity`, and `even-coverage`. A policy of `none` means addresses are not distributed automatically.

The assignment of addresses to a Network Interface Controller (NIC) can be done by using one of these two modes:

- CES legacy mode

  An address with the same subnet mask as the CES addresses, called the CES Base Address, is defined on each CES node. CES addresses are assigned to a NIC where the CES Base Address is hosted. In this mode, only IPv4 addresses are supported and no other non-CES address with the same subnet mask as the CES Base Address should exist on CES nodes.

- CES interface mode

  The customer defines one or more individual NICs on each CES node. The CES addresses are assigned to these NICs. To manage subnets, like it is currently done for CES nodes, a NIC can be a member of one or more CES groups. In this case a CES address can only be assigned to a NIC if the NIC is in the same group as the CES address. In this mode IPv6 and IPv4 addresses are supported.

Note: The address balancing done by policies works node-wide. This means that if a CES address can be hosted on different NICs on the same node, the NIC where the address is hosted is not defined. Well-defined address assignment on a node with multiple NICs is achieved by the usage of CES group.

Transitioning from legacy mode to interface mode: In a legacy mode, no IPv6 addresses can be added. You can switch from the legacy mode to the interface mode only if you have a valid NIC configuration. To switch from a legacy mode to an interface mode, add the NIC first and then switch the CES mode to interface. When the NICs are specified in the legacy mode you are in transition. This state is allowed but not recommended. You must switch to the interface mode, or remove the NIC configuration by using the **mmces interface mode interface** command.

Transitioning from interface mode to legacy mode: Use the **mmces interface mode legacy --force** command to switch back to the legacy mode while keeping the NIC assignment. The **mmces interface mode legacy --force** command deletes the NIC configuration and sets the cluster in legacy mode. Transitioning from the interface mode to the legacy mode unassigns all the IPv6 addresses. If you do not have CES Base Addresses, all the IPv4 addresses are also unassigned.

Use the **mmces interface** command to manage CES modes and NIC assignment.

A CES node can be placed in a suspended state. When a node is in suspended state, all of the CES addresses for that node are reassigned to other nodes, or set to unassigned. The node will not accept new address assignments. Any services that are started when the node is suspended continue to run, except `mmcesmonitord`. The `mmcesmonitord` service is stopped, and the lock on the `sharedRoot` file system is released. The `sharedRoot` file system is still mounted, but can be unmounted. The suspended state is persistent, which means nodes remain suspended following a reboot. After a reboot no services are running on a suspended node. However, if a node is not suspended, the services that were enabled on the nodes are restarted after the reboot.

The presence of a CES IP on a name node in an HDFS group activates the HDFS service. When there are multiple name nodes in an HDFS group, only the name node which has a CESIP assigned to it can be active. The absence of a CES IP deactivates the HDFS service from that name node. An HDFS group has the prefix `hdfs` followed by at least one alphabetical or numeric character. For example, `hdfsbda1`. You can assign a group or an attribute to a CES IP. If an HDFS group is assigned or unassigned from a CES IP, the related attribute is also assigned or unassigned automatically. A CES IP assigned to an HDFS group must not have any attributes other than the ones automatically assigned. An HDFS group must be assigned only once to all the CES IPs in the cluster.

Parameters

service
Manages protocol services with one of the following actions:

enable
Enables and starts the specified service on all CES nodes.

disable
Disables and stops the specified service on all CES nodes.

Note: Disabling a service will discard any configuration data from the CES cluster and needs to be used with caution. If applicable, backup any relevant configuration data. Subsequent service enablement will start with a clean configuration.

--force
Allows the `service disable` action to go through without confirmation from the user.

start
Starts the specified service on the nodes specified. If neither the `-N` or `-a` parameters are specified, the service is started on the local node.

stop
Stops the specified service on the nodes specified. If neither the `-N` or `-a` parameters are specified, the service is stopped on the local node.

Note: If a service is stopped on a node that has CES addresses assigned, clients will not be able to access the service using any of the addresses assigned to that node. Access to the data from clients is not possible any more for services that are stopped. This state is not persistent, so after a reboot all the services become active again.

list
Lists the state of the enabled services.

node
Manages CES node state with one of the following actions:

suspend
Suspends the specified nodes. If neither the `-N` or `-a` parameters are specified, only the local node is suspended. The `-a` stands for all CES nodes.

When a node is suspended, all addresses assigned to the node are reassigned to other nodes, or set to unassigned. The node will not accept any subsequent address assignments. Suspending a node triggers CES protocol recovery if the node has CES addresses assigned. This does not stop any services or the mmces monitor daemon.

If the `--stop` option is added to the suspend command, all the CES services including the mmces monitor daemon are stopped. This releases the lock on the `sharedRoot` file system, and it can be unmounted without further actions. The `sharedRoot` file system is still mounted, but can be unmounted.

Note: After a reboot no services are running on a suspended node. However, if a node is not suspended, the services that were enabled on the nodes are restarted after the reboot.

resume
Resumes the node so that CES IPs can be retrieved and starts the mmces monitor daemon if it is not running.

Using the `--start` option starts all the enabled protocols. The mmces monitor daemon is also started, if it is not already running. However, if at least one protocol was not successfully started, then all the protocols that were already running are also stopped.

list
Lists the specified nodes along with their current node state. If the `-N` parameter is not specified, all nodes are listed.

verbose
Lists the addresses assigned to the nodes.

--ces-group
Lists the nodes belonging to the specified groups.

log level
Sets the CES log level, which determines the amount of CES related information that is logged in the file, `/var/adm/ras/mmfs.log.latest`. The log level values can range from `0`, which is the default value, to 3. Increasing the log level adds to the information being logged.

The values are defined as follows:

level 0
Logs only non-repeated errors and non-repeated warnings. The non-repeated warnings indicate faulty behavior. Information about the state of an action is also logged during startup and shutdown.

level 1
Logs all errors and all non-repeated warnings.

level 2
Logs all warnings.

level 3
Logs all important debug information.

Important: A higher log level logs the data of all the lower log levels. Therefore, log level 1 includes the information of log level 0, log level 2 includes the information of log level 1 and log level 0, and so on.

The following status information is also accumulated in the log:

error
When an event that degrades the functionality of the system has occurred.

warning
When an unexpected event has occurred. However, this event does not degrade the functionality of the system. For example, an error was solved by a retry.

information
Documents the result of an operation. This is logged during startup and shutdown or in log level 3.

address
Manages CES addresses with one of the following actions:

add
Adds the addresses specified by the `--ces-ip` parameter to the CES address pool and assigns them to a node. The node to which an address is assigned will configure its network interfaces to accept network communication destined for the address. CES addresses must be different from IP addresses used for GPFS or CNFS communication.

If `--ces-node` is specified with `add`, all addresses specified with the `--ces-ip` parameter will be assigned to this node. If `--ces-node` is not specified, the addresses will be distributed among the CES nodes.

If an attribute is specified with `--attribute`, there can only be one address specified with the `--ces-ip` parameter.

If `--ces-group` is specified with `add`, all new addresses will be associated with the specified group. The result can be viewed with the **mmces address list** command.

Note: The provided addresses or host names must be resolvable by forward and reverse name resolution (DNS or `/etc/hosts` on all CES nodes). Otherwise you get the following error message:

```
Cannot resolve <ip address>; Name or service not known
```

You can also perform a manual check by running the following command: **mmcmi host <ip address>**.

Ensure that the netmask (PREFIX) setting in the `ifcfg-<interface>` files is correct.

remove
Removes the addresses specified by the `--ces-ip` parameter from the CES address pool. The node to which the address is assigned reconfigures its network interfaces to no longer accept communication for that address.

move
Moves addresses.

If the `--ces-ip` parameter is specified, the addresses specified by *IP* are moved from one CES node to another. The addresses are reassigned to the node specified by the `--ces-node` parameter.

If the `--rebalance` parameter is specified, the addresses are distributed within 60 seconds based on the currently configured distribution policy. If the policy is currently undefined or `none`, the `even-coverage` policy is applied.

Note: The information about the address movement is printed after the rebalance is done. The address movement is also done in the background periodically.

Use this command with caution because IP movement will trigger CES protocol recovery.

change
Changes or removes address attributes.

If the `--ces-ip` parameter is specified:

- The command associates the attributes that are specified by the `--attribute` parameter with the address that is specified by the `--ces-ip` parameter. If an attribute is already associated with another address, that association is ended.
- If the `--remove-attribute` parameter is specified, the command removes the attributes that are specified by the `--attribute` parameter from the addresses that are specified by the `--ces-ip` parameter.
- The command associates the groups that are specified by the `--ces-group` parameter with the address that is specified by the `--ces-ip` parameter
- If the `--remove-group` parameter is specified, the command removes the groups that are specified by the `--ces-group` parameter from the addresses that are specified by the `--ces-ip` parameter.

If the `--ces-ip` parameter is not specified:

- If the `--remove-attribute` parameter is specified, the command removes the attributes that are specified by the `--attribute` parameter from their current associations.
- If the `--remove-group` parameter is specified, the command removes the groups that are specified by the `--group` parameter from their current associations.
- Specifying `--remove-group` with the groups specified by the `--group` parameter removes the groups from their current associations.

list
Lists the CES addresses along with group, attribute and node assignments.

Options:

`--ces-ip` List only the addresses provided.

`--ces-group` List only addresses whose group assignment matches one of the groups provided.

`--attribute` List only addresses whose attributes match one of the attributes provided.

`--by-node` List addresses by node, using the output format from IBM Spectrum Scale 4.1.1 and later.

`--extended-list` Lists the preferred node of the given address in a new column if the `address balancing mode` option is set to `node-affinity`.

`--full-list` Lists the information about the preferred node and the node names where the given address could not be hosted in two new columns.

Note: The `[-N {Node[,Node...]| NodeFile | NodeClass| -a]` option is deprecated for IBM Spectrum Scale 4.2.3 and might be removed in a later release.

policy
Sets the CES address distribution policy.

prefix-ipv4
Sets the default prefix value for the IPv4 address or shows this value.

prefix-value
The value to which you want to set the default prefix for the IPv4 address. The *prefix-value* can be any number from 1 to 30.

show
Displays the IPv4 prefix value.

state
Shows the state of one or more nodes in the cluster.

show
Shows the state of the specified service on the nodes specified. If no service is specified, all services will be displayed. If neither the `-N` or `-a` parameters are specified, the state of the local node is shown.

cluster
Shows the combined state for the services across the whole CES cluster. If no service is specified an aggregated state will be displayed for each service, where healthy means the service is healthy on all nodes, degraded means the service is not healthy on one or all nodes, and failed means that the service is not available on any node. If a service is specified the state of that service will be listed for each node, along with the name of any event that is contributing to an unhealthy state.

--ces-node
Indicates that the command applies only to the specified CES node name.

--attribute
Specifies either a single attribute or a comma-separated list of attributes as indicated in the command syntax.

--ces-ip
Specifies either a single or comma-separated list of DNS qualified host names or IP addresses as indicated in the command syntax.

--rebalance
Distributes addresses immediately based on the currently configured distribution policy. If the policy is currently undefined or `none`, the `even-coverage` policy is applied.

none
Specifies that addresses are not distributed automatically.

balanced-load
Distributes addresses dynamically in order to approach an optimized load distribution throughout the cluster. The network and CPU load on all the nodes is monitored and addresses are moved based on given policies.

Addresses that were recently moved or addresses with attributes are not moved.

node-affinity
Attempts to keep addresses associated with the node to which they were assigned. Address node associations are created with the `--ces-node` parameter of the `mmces address add` command or the `mmces address move` command. Automatic movements of addresses do not change the association. Addresses that were enabled without a node specification do not have a node association. Addresses that are associated with a node but assigned to a different node are moved back to the associated node.

Addresses that were recently moved or addresses with attributes are not moved.

even-coverage
Attempts to evenly distribute all of the addresses among the available nodes.

Addresses that were recently moved or addresses with attributes are not moved.

--remove-attribute
Indicates that the specified attributes should be removed.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Indicates that the command applies only to the specified node names.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

-a
Specifies that the command applies to all CES nodes.

NFS
Specifies that the command applies to the NFS service.

OBJ
Specifies that the command applies to the Object service.

SMB
Specifies that the command applies to the SMB service.

BLOCK
Specifies that the command applies to the BLOCK service.

AUTH
Specifies that the command applies to the AUTH service.

NETWORK
Specifies that the command applies to the NETWORK service.

CES
Specifies that the command applies to the CES service.

--verbose
Specifies that the output is verbose.

new-level
Sets the log level to a new value. If the *new-level* parameter is not specified, the current log level is displayed.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

--time
Lists the previous events from one of the following intervals:

hour
Lists the events from the past hour.

day
Lists the events from the past day.

week
Lists the events from the past week.

month
Lists the events from the past month.

The events are listed whether or not they are currently contributing to the state of a component.

interface
Shows the state of one or more nodes in the cluster.

mode
Manage the CES mode.

legacy
Set the cluster in CEC legacy mode. IPv6 support is disabled.

Address assignment is done by CES Base Addresses. If the command is executed in the interface mode it is rejected, because CES cannot verify if CES Base Addresses are assigned.

interface
Set the cluster in CES interface mode and supports IPv6 addresses.

Address assignment is done by NIC configuration. The command is rejected if problems occur in the CES address assignments when this command is executed.

--clear
Remove the CES NIC configuration and set the cluster to legacy mode. This command requests user validation.

--force
Overwrites any check or request for user validation.

list
Shows the CES mode. The CES mode is either legacy or interface.

--by-node
Displays the CES mode for the specified node or nodes.

--by-group
Displays the CES mode for the specified group or groups.

ALL

Display all the NICs with at least one group assigned to them.

ANY
Displays all the NICs.

NONE
Displays all the NICs with no group assigned to them.

add
Add specified NICs to the cluster using the specified options. The command is rejected if problems occur in the CES address assignments when this command is executed.

--nic
Specify the NICs as a comma-separated list.

--ces-group
Sets all the NICs specified in this command into the specified group or groups. Groups are specified by a comma-separated list.

--ces-node
Limit the NIC assignment to the specified node or nodes. Nodes are specified by a comma-separated list.

--force
Disable checking for potential problems.

remove
Remove the specified NICs from the cluster using the specified options. The command is rejected if problems occur in the CES address assignments when this command is executed.

--nic
Specify the NICs as a comma-separated list.

--ces-node
Limit the NIC removal to the specified node or nodes. Nodes are specified by comma-separated list.

--force
Disable checking for potential problems.

check

Checks the NIC configuration against the node groups and CES addresses.

If the NIC configuration matches the node groups the configuration is valid. The command checks if any NIC is defined, and if there is any CES address that cannot be assigned to that NIC. You can check if a CES address cannot be assigned to a NIC by verifying if any of the NIC groups is assigned to a CES address. No checks are done for suspended nodes or for NICs that are currently down.

events
Shows one of the following CES events that occurred on a node or nodes:

active
Lists all events that are currently contributing to making the state of a component unhealthy. If no component is specified, active events for all components are listed. If neither the `-N` or `-a`

parameters are specified, the active events for the local node are listed. If there are multiple events shown by the command they will be listed in the order we recommend they be fixed, with the most important event to fix at the top.

list
Lists the events that occurred on a node or nodes, whether or not they are currently contributing to the state of a component. If no component is specified, events for all components are listed. If `--time` is specified, only events from the previous chosen interval are listed, otherwise all events are listed. If `--severity` is specified, only events of the chosen severity are listed, otherwise all events are listed. If neither the `-N` or `-a` parameters are specified, the events for the local node are listed.

Events older than 180 days are removed from the list. A maximum of 10,000 events are saved in the list.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmces` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To add an address to a specified node, issue the following command:

```
# mmces address add --ces-node node1 --ces-ip 10.1.2.3
```

   When this command is successful, the system does not display output.
2. To add several addresses to a specified node, issue the following command:

```
# mmces address add --ces-node node1 --ces-ip 10.1.2.3,10.1.2.4
```

   When this command is successful, the system does not display output.
3. To add an address with the attribute `object_singleton_node` to a specified node, issue the following command:

```
# mmces address add --ces-node node1 --ces-ip 10.1.2.3 --attribute object_singleton_node
```

   When this command is successful, the system does not display output.
4. To add addresses which are distributed among the CES nodes, issue the following command:

```
# mmces address add --ces-ip 10.1.2.3,10.1.2.4,10.1.2.5,10.1.2.6
```

   When this command is successful, the system does not display output.
5. To remove several addresses, issue the following command:

```
# mmces address remove --ces-ip 10.1.2.3,10.1.2.4
```

   When this command is successful, the system does not display output.

6. To associate the attribute object_singleton_node to the address 10.1.2.3, issue the following command:

```
# mmces address change --ces-ip 10.1.2.3 --attribute object_singleton_node
```

When this command is successful, the system does not display output.

7. To remove the attribute object_singleton_node, issue the following command:

```
# mmces address change --remove-attribute --attribute object_singleton_node
```

When this command is successful, the system does not display output.

8. To move an address to another node, issue the following command:

```
# mmces address move --ces-ip 10.0.100.231 --ces-node node2
```

When this command is successful, the system does not display output.

9. To set a default prefix for the IPv4 address, issue the following command:

```
# mmces address prefix-ipv4 30
```

A sample output is as follows:

```
CES CESNETMASK_IPv4 is 30
```

10. To suspend a group of nodes, issue the following command:

```
# mmces node suspend -N node1,node2,node3
```

A sample output is as follows:

```
Node now in suspended state.
```

11. To resume the current node, issue the following command:

```
# mmces node resume
```

A sample output is as follows:

```
Node no longer in suspended state.
```

12. To resume a node and start all enabled CES services, issue the following command:

```
# mmces node resume --start
```

13. To suspend a node, issue the following command:

```
# mmces node suspend
```

14. To suspend a node and stop all CES services, issue the following command:

```
# mmces node suspend --stop
```

15. To enable the Object service in the CES cluster, issue the following command:

```
# mmces service enable obj
```

When this command is successful, the system does not display output.

16. To disable the NFS service in the CES cluster, issue the following command:

```
# mmces service disable nfs
```

When this command is successful, the system does not display output.

17. To stop the SMB service on a few nodes, issue the following command:

```
# mmces service stop smb -N node1,node2,node3
```

When this command is successful, the system does not display output.

18. To start the SMB service on all CES nodes, issue the following command:

```
# mmces service start smb -a
```

When this command is successful, the system does not display output.

19. To show which services are enabled and which are running all CES nodes, issue the following command:

```
# mmces service list -a
```

A sample output is as follows:

```
Enabled services: NFS OBJ
node1: NFS is running, OBJ is running
node2: NFS is running, OBJ is running
node3: NFS is running, OBJ is running
```

20. To display the current CES log level, issue the following command:

```
# mmces log level
```

A sample output is as follows:

```
CES log level is currently set to 1
```

21. To set the CES log level to 2, issue the following command:

```
# mmces log level 2
```

A sample output is as follows:

```
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all affected nodes.
This is an asynchronous process.
```

22. To display the state of all CES components on the local node, issue the following command:

```
# mmces state show
```

A sample output is as follows:

```
NODE    AUTH         BLOCK        NETWORK      AUTH_OBJ     NFS          OBJ          SMB          CES
node1   DISABLED     DISABLED     HEALTHY      DISABLED     DISABLED     DISABLED     HEALTHY      HEALTHY
```

23. To display the state of the NFS component on all nodes, issue the following command:

```
# mmces state cluster NFS
```

A sample output is as follows:

```
NODE   COMPONENT  STATE    EVENTS
node1  NFS        HEALTHY
node2  NFS        FAILED   nfsd_down
node3  NFS        HEALTHY
```

24. To display a list of active events of all CES components on the local node, issue the following command:

```
# mmces events active
```

A sample output is as follows:

```
Node    Component   Event Name   Severity     Details
node1   NFS         nfsd_down    ERROR        NFSD process not running
```

25. To display a list of all NFS events from the last hour on the local node, issue the following command:

```
# mmces events list
```

A sample output is as follows:

```
Node  Timestamp                          Event Name              Severity Details
node1 2015-05-13 10:57:52.124369+00:00UTC nfsd_down               ERROR    NFSD process not running
node1 2015-05-13 10:58:06.809071+00:00UTC cesnodestatechange_info INFO     A CES node state changed
node1 2015-05-13 10:58:07.137343+00:00U   ganeshagrace_info       INFO     Ganesha NFS is set to grace
```

26. To enable the block service, issue the following command:

```
# mmces service enable BLOCK
```

The system displays the following prompt:

```
Block device support in Spectrum Scale is intended for use only in diskless node
remote boot (non-performance-critical), and is not suited for high-bandwidth
block device access needs. Confirm that this matches your use case before enabling
the block service. If you have any questions contact scale@us.ibm.com
Do you want to continue to enable BLOCK service? (yes/no)
```

After you press Y, the system displays the following output:

```
c40bbc1xn12.gpfs.net: Loading and configuring SCST
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

27. To add a NIC to all CES nodes in the cluster, run the following command:

```
# mmces interface add --nic eth1
```

The system displays the following prompt:

```
Accepted: 3 Skipped: 0

Additional messages:

Accepted changes:
Node   Node Name                accepted Nic's   accepted Nic's groups
------ ------------------------ ---------------- ----------------------
2      cluster-22.localnet.com   eth1
3      cluster-23.localnet.com   eth1
4      cluster-24.localnet.com   eth1
```

28. To add a NIC to selected nodes in the cluster with CES group assignment, run the following command:

```
# mmces interface add --nic eth1 --ces-node cluster-23,cluster-24 --ces-group ipv6
```

The system displays the following prompt:

```
Accepted: 2 Skipped: 0

Additional messages:

Accepted changes:
Node   Node Name                accepted Nic's   accepted Nic's groups
------ ------------------------ ----------------
----------------------
3      cluster-23.localnet.com   eth1             ipv6
4      cluster-24.localnet.com   eth1             ipv6
```

29. To remove the CES NIC configuration, run the following command:

```
# mmces interface mode legacy --clear
```

The system displays the following prompt:

```
Debug: legacy
The CES NIC configuration will be removed and IPv6 addresses are not supported anymore.
CES-Base-IPs will be needed to define NICs where CES IPs are hosted.
Type 'yes' to continue. Any other input will abort the command.
yes
The CES NIC configuration is removed. Cluster is in legacy mode using CES base IPs for IPv4
only.
```

30. To verify the current configuration, run the following command:

```
# mmces interface check
```

The system displays the following prompt:

```
Existing NIC configuration is valid, and can host all CES IPs.
```

31. To switch to interface mode, run the following command:

```
# mmces interface mode interface
```

The system displays the following prompt:

```
Set CES mode to interface

Additional messages:
CES interface mode: IPv6 support enabled. Address assignement by user defined NIC's.
```

See also

- “mmchconfig command” on page 170
- “mmlscluster command” on page 492
- “mmlsconfig command” on page 495
- “mmnfs command” on page 560
- “mmobj command” on page 573
- “mmsmb command” on page 709
- “mmuserauth command” on page 738

Location

/usr/lpp/mmfs/bin

mmcesdr command

Manages protocol cluster disaster recovery.

Synopsis

```
mmcesdr primary config --output-file-path FilePath --ip-list IPAddress[,IPAddress,...]
                       [--allowed-nfs-clients {--all | --gateway-nodes |
IPAddress[,IPAddress,...]}]
                       [--rpo RPOValue] [--inband] [-v]
```

or

```
mmcesdr primary backup [-v]
```

or

```
mmcesdr primary restore [--new-primary] [--input-file-path FilePath]
                        [--file-config {--recreate | --restore}] [-v]
```

or

```
mmcesdr primary update {--obj | --nfs | --smb | --ces} [-v]
```

or

```
mmcesdr primary failback --prep-outband-transfer --input-file-path FilePath [-v]
```

or

```
mmcesdr primary failback --convert-new --output-file-path FilePath --input-file-path FilePath
[-v]
```

or

```
mmcesdr primary failback {--start | --apply-updates | --stop [--force]} [--input-file-path
FilePath] [-v]
```

or

```
mmcesdr secondary config --input-file-path FilePath [--prep-outband-transfer] [--inband] [-v]
```

or

```
mmcesdr secondary failover [--input-file-path FilePath][--file-config {--recreate | --restore}]
                           [ --data {--restore | --norestore}] [-v]
```

or

```
mmcesdr secondary failback --generate-recovery-snapshots --output-file-path FilePath
                           [--input-file-path FilePath] [-v]
```

or

```
mmcesdr secondary failback --post-failback-complete [--input-file-path FilePath]
                           [--file-config {--recreate | --restore}][-v]
```

or

```
mmcesdr secondary failback --post-failback-complete --new-primary --input-file-path FilePath
                           [--file-config {--recreate | --restore}] [-v]
```

Availability

Available with IBM Spectrum Scale Advanced Edition, IBM Spectrum Scale Data Management Edition, IBM Spectrum Scale Developer Edition, or IBM Spectrum Scale Erasure Code Edition.

Description

Use the `mmcesdr` command to manage protocol cluster disaster recovery.

You can use the **mmcesdr primary config** command to perform initial configuration for protocols disaster recovery on the primary cluster and to generate a configuration file that is used on the secondary cluster. The protocol configuration data can be backed up using the **mmcesdr primary backup** command and the backed-up data can be restored using the **mmcesdr primary restore** command. The backed-up configuration information for the primary cluster can be updated by using the **mmcesdr primary update** command. You can use the **mmcesdr primary failback** command to fail back the client operations to the primary cluster.

You can use the **mmcesdr secondary config** command to perform initial configuration for protocols disaster recovery on the secondary cluster by using the configuration file generated from the primary cluster. The secondary read-only filesets can be converted into read-write primary filesets using the **mmcesdr secondary failover** command. You can use the **mmcesdr secondary failback** command to either generate a snapshot for each acting primary fileset or complete the failback process, and convert the acting primary filesets on the secondary cluster back into secondary filesets.

For information on detailed steps for protocols disaster recovery, see *Protocols cluster disaster recovery* in *IBM Spectrum Scale: Administration Guide*.

The **mmcesdr** log file is at `/var/adm/ras/mmcesdr.log`. This log file is included with the CES information generated by the **gpfs.snap** command. The **gpfs.snap** command generates the CES information by default, if a protocol is enabled.

Parameters

primary
This command is run on the primary cluster.

config
Perform initial configuration of protocol cluster disaster recovery.

--output-file-path ***FilePath***
Specifies the path to store output of the generated configuration file, which is always named `DR_Config`.

--ip-list ***IPAddress[,IPAddress,...]***
Comma-separated list of public IP addresses on the secondary cluster to be used for active file management (AFM) DR-related NFS exports.

--allowed-nfs-clients ***{--all | --gateway-nodes | IPAddress[,IPAddress,...]}***
Optional. Specifies the entities that can connect to the AFM DR-related NFS shares, where:

--all
Specifies that all clients must be allowed to connect to the AFM DR-related NFS shares. If omitted, the default value of *--all* is used.

--gateway-nodes
Specifies the gateway nodes currently defined on the primary that must be allowed to connect to the AFM DR-related NFS shares.

IPAddress[,IPAddress,...]
Specifies the comma-separated list of IP addresses that must be allowed to connect to the AFM DR-related NFS shares.

--rpo *RPOValue*

Optional. Specifies the integer value of the recovery point objective (RPO) to be used for the AFM DR filesets. By default, this parameter is disabled. The valid range is: 720 <= RPO <= 2147483647. The minimum value that can be set for the RPO is 720.

Note: Setting the value of RPO to less than 720 generates an error.

--inband
Optional. Specifies to use the inband (across the WAN) method of initial data transfer from primary to secondary cluster. If omitted, the default value of outband is used.

backup
Backs up all protocol configuration and CES configuration into a dedicated, independent fileset with each protocol in its own subdirectory.

restore
Restores object, NFS, and SMB protocol configuration and CES configuration from the configuration data backed up.

--new-primary
Optional. Performs restore operation to a newly, failed back primary cluster.

--input-file-path *FilePath*

Optional. Specifies the original configuration file that was used to set up the secondary cluster. If not specified, the file that is saved in the configuration independent fileset is used as default.

--file-config {--recreate | --restore}

Optional. Specifies whether SMB and NFS exports are re-created, or if the entire protocol configuration is restored. If not specified, the SMB and NFS exports are re-created by default.

update
Updates the backed-up copy of the protocol configuration or CES configuration.

--obj
Specifies the backed up-copy of the object protocol configuration to be updated with the current object configuration.

--nfs
Specifies the backed-up copy of the IBM NFSv4 stack protocol configuration to be updated with the current IBM NFSv4 stack configuration.

--smb
Specifies the backed-up copy of the SMB protocol configuration to be updated with the current SMB configuration.

--ces
Specifies the backed-up copy of the CES configuration to be updated with the current CES configuration.

failback

Used for several options to failback client operations to a primary cluster.

Failback involves transfer of data from the acting primary (secondary) cluster to the old primary cluster as well as restoring protocol and possibly CES configuration information and transformation of protected filesets to primary filesets.

--prep-outband-transfer
Creates independent filesets that out of band data is transferred to.

--input-file-path *FilePath*
Specifies the configuration file that is the output from the **mmcesdr secondary failback --generate-recovery-snapshots** command.

failback

Used for several options to failback client operations to a primary cluster.

Failback involves transfer of data from the acting primary (secondary) cluster to the old primary cluster as well as restoring protocol and possibly CES configuration information and transformation of protected filesets to primary filesets.

--convert-new
Specifies that the failback is not going to the old primary but instead a new primary. This step specifically converts the newly created independent filesets to primary AFM DR filesets.

--output-file-path *FilePath*
Specifies the path to store output of generated configuration file, `DR_Config`, with the new AFM primary IDs.

--input-file-path *FilePath*
Specifies the configuration file that is the output from the **`mmcesdr secondary failback --generate-recovery-snapshots`** command.

failback

Used for several options to failback client operations to a primary cluster.

Failback involves transfer of data from the acting primary (secondary) cluster to the old primary cluster as well as restoring protocol and possibly CES configuration information and transformation of protected filesets to primary filesets.

--start
Begins the failback process and restores the data to the last RPO snapshot.

--apply-updates
Transfers data that was written to the secondary cluster while failover was in-place.

Note: The use of this option might need to be done more than once depending on the system load.

--stop [--force]
Completes the transfer of data process and puts the filesets in the read-write mode. Optionally, if this fails you can use the `--force` option.

Note: In addition to using these options, after stopping the data transfer, you need to use the **`mmcesdr primary restore`** command to restore the protocol and the CES configuration.

--input-file-path *FilePath*
Optional. Specifies the original configuration file that was used to set up the secondary cluster. If not provided, the default is to use the one saved in the configuration independent fileset.

secondary
This command is run on the secondary cluster.

config
Perform initial configuration of protocol cluster disaster recovery.

--prep-outband-transfer
Creates independent filesets that out of band data is transferred to as part of the initial configuration. If out of band data transfer is used for DR configuration, this option must be used before data is transferred from the primary to the secondary using out of band transfer. If out of band transfer is used, this command is run once with this option and then again after the data is transferred without the option.

--input-file-path *FilePath*
Specifies the path of the configuration file generated from the configuration step of the primary cluster.

--inband
Optional. Specifies to use the inband (across the WAN) method of initial data transfer from primary to secondary cluster. If omitted, the default value of outband is used.

Note: If `--inband` is used for the primary configuration, it must also be used for the secondary configuration.

failover
Converts secondary filesets from read-only to read-write primary filesets and converts the secondary protocol configurations to those of the failed primary.

--input-file-path *FilePath*
Optional. Specifies the original configuration file that was used to set up the secondary cluster. If not specified, the file that is saved in the configuration independent fileset is used as default.

--file-config {--recreate | --restore}
Optional. Specifies whether SMB and NFS exports are re-created, or if the entire protocol configuration is restored. If not specified, the SMB and NFS exports are re-created by default.

--data {--restore | --norestore}
Optional. Specifies that data must be restored from the latest RPO snapshot or that restoring from the latest RPO snapshot is not required. Default is that restoring from the latest RPO snapshot is not required.

failback
Runs one of the two failback options: either generates a snapshot for each acting primary fileset or completes the failback process and convert the acting primary filesets on the secondary cluster back into secondary filesets

--generate-recovery-snapshots
Generates the `psnap0` snapshot for each acting primary fileset and stores in the default snapshot location for use in creation of a new primary cluster with new primary filesets to fail back to. The files within the snapshot need to be manually transported to the new primary.

--output-file-path *FilePath*
Specifies the path to store output of generated snapshot recovery configuration file.

--input-file-path *FilePath*
Optional. Specifies the path of the original configuration file that was used to set up the secondary cluster. If not provided, the default is to use the one saved in the configuration independent fileset.

failback
Runs one of the two failback options: either generates a snapshot for each acting primary fileset or completes the failback process and convert the acting primary filesets on the secondary cluster back into secondary filesets

--post-failback-complete
Completes the failback process by converting the acting primary filesets back into secondary, read-only filesets and ensures that the proper NFS exports for AFM DR exist.

--new-primary
Performs the failback operation to a newly, failed back primary cluster.

--input-file-path *FilePath*
Specifies the path of the updated configuration file that is created from the **`mmcesdr primary failback --convert-new`** command, which includes updated AFM primary IDs.

--file-config {--recreate | --restore}

Optional. Specifies whether SMB and NFS exports are re-created, or if the entire protocol configuration is restored. If not specified, the SMB and NFS exports are re-created by default.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmcesdr` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see the topic *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

Examples

1. Issue the following command on the primary cluster to configure independent fileset exports as AFM DR filesets and backup configuration information:

```
mmcesdr primary config --output-file-path /root/ --ip-list "9.11.102.211,9.11.102.210" --rpo 720 --inband
```

   The system displays output similar to this:

```
Performing step 1/5, configuration fileset creation/verification.
Successfully completed step 1/5, configuration fileset creation/verification.
Performing step 2/5, protocol and export services configuration backup.
Successfully completed step 2/5, protocol and export services configuration backup.
Performing step 3/5, determination of protocol exports to protect with AFM DR.
WARNING: Export /gpfs/fs0/nfs-ganesha-dep of type nfs will NOT be protected through AFM DR because it is
a dependent fileset.
Not all exports of type NFS-ganesha will be protected through AFM DR, rc: 2
WARNING: Export /gpfs/fs0/smb-dep of type smb will NOT be protected through AFM DR because it is a
dependent fileset.
Not all exports of type SMB will be protected through AFM DR, rc: 2
Completed with errors step 3/5, determination of protocol exports to protect with AFM DR.
Performing step 4/5, conversion of protected filesets into AFM DR primary filesets.
Successfully completed step 4/5, conversion of protected filesets into AFM DR primary filesets.
Performing step 5/5, creation of output DR configuration file.
Successfully completed step 5/5, creation of output DR configuration file.

File to be used with secondary cluster in next step of cluster DR setup: /root//DR_Config
```

2. Issue the following command on the secondary cluster to create the independent filesets that are a part of the pair of AFM DR filesets associated with those on the primary cluster:

```
mmcesdr secondary config --input-file-path /root/ --inband
```

   In addition to fileset creation, this command also creates the necessary NFS exports and converts the independent filesets to AFM DR secondary filesets.

   The system displays output similar to this:

```
Performing step 1/3, creation of independent filesets to be used for AFM DR.
Successfully completed step 1/3, creation of independent filesets to be used for AFM DR.
Performing step 2/3, creation of NFS exports to be used for AFM DR.
Successfully completed step 2/3, creation of NFS exports to be used for AFM DR.
Performing step 3/3, conversion of independent filesets to AFM DR secondary filesets.
Successfully completed step 3/3, conversion of independent filesets to AFM DR secondary filesets.
```

3. Issue the following command on the primary cluster to configure independent fileset exports as AFM DR filesets, back up configuration information, and facilitate outband data transfer.

   Note: The outband data transfer is the default method of data transfer from the primary cluster to the secondary cluster when AFM DR fileset relationships are first set up.

```
mmcesdr primary config --output-file-path /root/ --ip-list "9.11.102.211,9.11.102.210" --rpo 720
```

   The system displays output similar to this:

```
Performing step 1/5, configuration fileset creation/verification.
Successfully completed step 1/5, configuration fileset creation/verification.
Performing step 2/5, protocol and export services configuration backup.
Successfully completed step 2/5, protocol and export services configuration backup.
Performing step 3/5, determination of protocol exports to protect with AFM DR.
```

```
Successfully completed step 3/5, determination of protocol exports to protect with AFM DR.
Performing step 4/5, conversion of protected filesets into AFM DR primary filesets.
Successfully completed step 4/5, conversion of protected filesets into AFM DR primary filesets.
Performing step 5/5, creation of output DR configuration file.
Successfully completed step 5/5, creation of output DR configuration file.

File to be used with secondary cluster in next step of cluster DR setup: /root//DR_Config
```

4. Issue the following command on the secondary cluster to create the independent filesets that will later be paired with those on the primary cluster to form AFM DR pairs as part of failing back to a new primary cluster:

```
mmcesdr secondary config --input-file-path /root --prep-outband-transfer
```

The system displays output similar to this:

```
Creating independent filesets to be used as recipients of AFM DR outband transfer of data.
Transfer all data on primary cluster for fileset fs0:combo1 to fileset fs0:combo1 on secondary cluster.
Transfer all data on primary cluster for fileset fs0:combo2 to fileset fs0:combo2 on secondary cluster.
Transfer all data on primary cluster for fileset fs0:nfs-ganesha1 to fileset fs0:nfs-ganesha1 on secondary cluster.
Transfer all data on primary cluster for fileset fs0:nfs-ganesha2 to fileset fs0:nfs-ganesha2 on secondary cluster.
Transfer all data on primary cluster for fileset fs0:smb1 to fileset fs0:smb1 on secondary cluster.
Transfer all data on primary cluster for fileset fs0:smb2 to fileset fs0:smb2 on secondary cluster.
Transfer all data on primary cluster for fileset fs1:async_dr to fileset fs1:async_dr on secondary cluster.
Transfer all data on primary cluster for fileset fs1:obj_sofpolicy1 to fileset fs1:obj_sofpolicy1 on secondary cluster.
mmcesdr: CES Object protocol is not enabled but there is an object related export present.
Skipping clearing out the object related files and directories from export.
Transfer all data on primary cluster for fileset fs1:obj_sofpolicy2 to fileset fs1:obj_sofpolicy2 on secondary cluster.
mmcesdr: CES Object protocol is not enabled but there is an object related export present.
Skipping clearing out the object related files and directories from export.
Transfer all data on primary cluster for fileset fs1:object_fileset to fileset fs1:object_fileset on secondary cluster.
mmcesdr: CES Object protocol is not enabled but there is an object related export present.
Skipping clearing out the object related files and directories from export.
Successfully completed creating independent filesets to be used as recipients of AFM DR outband transfer of data.
Transfer data from primary cluster through outbound trucking to the newly created independent filesets before
proceeding to the next step.
```

5. After all the data has been transferred to the secondary, issue the following command to complete the setup on the secondary:

```
mmcesdr secondary config --input-file-path /root
```

The system displays output similar to this:

```
Performing step 1/3, verification of independent filesets to be used for AFM DR.
Successfully completed step 1/3, creation of independent filesets to be used for AFM DR.
Successfully completed 1/3, verification of independent filesets to be used for AFM DR.
Performing step 2/3, creation of NFS exports to be used for AFM DR.
Successfully completed step 2/3, creation of NFS exports to be used for AFM DR.
Performing step 3/3, conversion of independent filesets to AFM DR secondary filesets.
Successfully completed step 3/3, conversion of independent filesets to AFM DR secondary filesets.
```

6. Issue the following command on the secondary cluster after the primary cluster has failed:

```
mmcesdr secondary failover
```

The system displays output similar to this:

```
Performing step 1/4, saving current NFS configuration to restore after failback.
Successfully completed step 1/4, saving current NFS configuration to restore after failback.
Performing step 2/4, failover of secondary filesets to primary filesets.
Successfully completed step 2/4, failover of secondary filesets to primary filesets.
Performing step 3/4, protocol configuration/exports restore.
Successfully completed step 3/4, protocol configuration/exports restore.
Performing step 4/4, create/verify NFS AFM DR transport exports.
Successfully completed step 4/4, create/verify NFS AFM DR transport exports.
```

7. Issue the following command on the secondary cluster to prepare recovery snapshots that contain data that is transferred to the new primary cluster:

```
mmcesdr secondary failback --generate-recovery-snapshots
--output-file-path "/root/" --input-file-path "/root/"
```

The system displays output similar to this:

```
Performing step 1/2, generating recovery snapshots for all AFM DR acting primary filesets.
Transfer all data under snapshot located on acting primary cluster at:
```

```
/gpfs/fs0/combo1/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-1 to
fileset link point of fileset fs0:combo1 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs0/combo2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-2 to
fileset link point of fileset fs0:combo2 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs0/nfs-ganesha1/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-3 to
fileset link point of fileset fs0:nfs-ganesha1 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs0/nfs-ganesha2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-4 to
fileset link point of fileset fs0:nfs-ganesha2 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs0/smb1/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-5 to
fileset link point of fileset fs0:smb1 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs0/smb2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-6 to
fileset link point of fileset fs0:smb2 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs1/.async_dr/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DECB-2 to
fileset link point of fileset fs1:async_dr on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs1/obj_sofpolicy1/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DECB-3 to
fileset link point of fileset fs1:obj_sofpolicy1 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs1/obj_sofpolicy2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DECB-4 to
fileset link point of fileset fs1:obj_sofpolicy2 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs1/object_fileset/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DECB-1 to
fileset link point of fileset fs1:object_fileset on new primary cluster.
Successfully completed step 1/2, generating recovery snapshots for all AFM DR acting
primary filesets.
Performing step 2/2, creation of recovery output file for failback to new primary.
Successfully completed step 2/2, creation of recovery output file for failback to new
primary.

File to be used with new primary cluster in next step of failback to new primary cluster: /
root//DR_Config
```

8. Issue the following command on the primary cluster to restore the protocol and export services configuration information:

   ```
   mmcesdr primary restore --new-primary
   ```

   The system displays output similar to this:

   ```
   Restoring cluster and enabled protocol configurations/exports.
   Successfully completed restoring cluster and enabled protocol configurations/exports.
   ```

9. Issue the following command on the secondary cluster to restore the protocol and export services configuration information:

   ```
   mmcesdr secondary failback --post-failback-complete --new-primary --input-file-path "/root"
   ```

   The system displays output similar to this:

   ```
   Performing step 1/2, converting protected filesets back into AFM DR secondary filesets.
   Successfully completed step 1/2, converting protected filesets back into AFM DR secondary filesets.
   Performing step 2/2, restoring AFM DR-based NFS share configuration.
   Successfully completed step 2/2, restoring AFM DR-based NFS share configuration.
   ```

10. Issue the following command on the primary cluster to back up configuration:

    ```
    mmcesdr primary backup
    ```

    The system displays output similar to this:

    ```
    Performing step 1/2, configuration fileset creation/verification.
    Successfully completed step 1/2, configuration fileset creation/verification.
    Performing step 2/2, protocol and export services configuration backup.
    Successfully completed step 2/2, protocol and export services configuration backup.
    ```

11. Issue the following command on the primary cluster to restore configuration when the primary cluster is not in a protocols DR relationship with another cluster:

```
mmcesdr primary restore --file-config --restore
```

The system displays output similar to this:

```
Restoring cluster and enabled protocol configurations/exports.
Successfully completed restoring cluster and enabled protocol configurations/exports.

================================================================================
= If all steps completed successfully, remove and then re-create file
= authentication on the Primary cluster.
= Once this is complete, Protocol Cluster Configuration Restore will be complete.
================================================================================
```

See also

- “mmafmctl command” on page 61
- “mmces command” on page 133
- “mmchconfig command” on page 170
- “mmlscluster command” on page 492
- “mmlsconfig command” on page 495
- “mmnfs command” on page 560
- “mmobj command” on page 573
- “mmpsnap command” on page 614
- “mmrestorefs command” on page 675
- “mmsmb command” on page 709
- “mmuserauth command” on page 738

Location

/usr/lpp/mmfs/bin

mmchattr command

Changes attributes of one or more GPFS files.

Synopsis

```
mmchattr [-m MetadataReplicas] [-M MaxMetadataReplicas]
         [-r DataReplicas] [-R MaxDataReplicas] [-P DataPoolName]
         [-D {yes | no}] [-I {yes | defer}] [-i {yes | no}]
         [-a {yes | no}] [-l]
         [{--set-attr AttributeName[=Value] [--pure-attr-create | --pure-attr-replace]} |
          {--delete-attr AttributeName [--pure-attr-delete]}]
         [--hex-attr] [--hex-attr-name] [--no-attr-ctime]
         [--compact[=[Option][,Option]...]]
         [--compression {yes | no | z | lz4 | zfast | alphae | alphah}]
         [--block-group-factor BlockGroupFactor]
         [--write-affinity-depth WriteAffinityDepth]
         [--write-affinity-failure-group "WadfgValueString"]
         [--indefinite-retention {yes | no}]
         [--expiration-time yyyy-mm-dd[@hh:mm:ss]]
         {--inode-number [SnapPath/]InodeNumber [[SnapPath/]InodeNumber...] |
          Filename [Filename...]}
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmchattr` command to change the replication attributes, storage pool assignments, retention and immutability attributes, I/O caching policy, file compression or decompression of files in the file system, and resolve data block replica mismatches using the `gpfs.readReplicaRule` extended attribute. For more information, see the *Replica mismatches* topic in the *IBM Spectrum Scale: Problem Determination Guide*.

The replication factor must be less than or equal to the maximum replication factor for the file. If insufficient space is available in the file system to increase the number of replicas to the value requested, the `mmchattr` command ends. However, replication factor for some blocks of the file might increase after the `mmchattr` command ends. If later (when you add another disk), free space becomes available in the file system you can then issue the `mmrestripefs` command with the `-r` or `-b` option to complete the replication of the file. The `mmrestripefile` command can be used in a similar manner. You can use the `mmlsattr` command to display the replication values.

Data of a file is stored in a specific storage pool. A storage pool is a collection of disks or RAIDs with similar properties. Because these storage devices have similar properties, you can manage them as a group. You can use storage pools to do the following tasks:

- Partition storage for the file system.
- Assign file storage locations.
- Improve system performance.
- Improve system reliability.

The Direct I/O caching policy bypasses file cache and transfers data directly from disk into the user space buffer, as opposed to using the normal cache policy of placing pages in kernel memory. Applications with poor cache hit rates or a large amount of I/O might benefit from the use of Direct I/O.

The `mmchattr` command can be run against a file in use.

You must have write permission for the files whose attributes you are changing unless you are changing the `gpfs.readReplicaRule` extended attribute and you are the file owner, in which case read permission is sufficient.

Parameters

-m *MetadataReplicas*
Specifies how many copies of the file system's metadata to create. Valid values are 1, 2, and 3. This value cannot be greater than the value of the *MaxMetadataReplicas* attribute of the file.

-M *MaxMetadataReplicas*
Specifies the maximum number of copies of indirect blocks for a file. Space is reserved in the inode for all possible copies of pointers to indirect blocks. Valid values are 1, 2, and 3. This value cannot be less than the value of the *DefaultMetadataReplicas* attribute of the file.

-r *DataReplicas*
Specifies how many copies of the file data to create. Valid values are 1, 2, and 3. This value cannot be greater than the value of the *MaxDataReplicas* attribute of the file.

-R *MaxDataReplicas*
Specifies the maximum number of copies of data blocks for a file. Space is reserved in the inode and indirect blocks for all possible copies of pointers to data blocks. Valid values are 1, 2, and 3. This value cannot be less than the value of the *DefaultDataReplicas* attribute of the file.

-P *DataPoolName*
Changes the assigned storage pool of the file to the specified *DataPoolName*. The caller must have superuser or root privileges to change the assigned storage pool.

-D {yes | no}
Enable or disable the Direct I/O caching policy for files.

-I {yes | defer}
Specifies whether replication and migration between pools, or file compression or decompression, is to be performed immediately (`-I yes`), or deferred until a later call to `mmrestripefs` or `mmrestripefile` (`-I defer`). By deferring the operation, you can complete it when the system is not loaded with processes or I/O. Also, if multiple files are affected, the data movement can be done in parallel. The default is `-I yes`. For more information about file compression and decompression, see the `--compression` option in this topic.

-i {yes | no}
Specifies whether the file is immutable (**-i yes**) or not immutable (**-i no**).

Note: The immutability attribute is specific to the current instance of the file. Restoring an image of the file to another location does not retain the immutability option. You must set it yourself.

-a {yes | no}
Specifies whether the file is in `appendOnly` mode (`-a yes`) or not (`-a no`).

Notes:

1. The `appendOnly` setting is specific to the current instance of the file. Restoring an image of the file to another location does not retain the `appendOnly` mode. You must set it yourself.
2. `appendOnly` mode is not supported for AFM filesets.

-l
Specifies that this command works only with regular files and directories and does not follow symlinks. The default is to follow symlinks.

--set-attr *AttributeName*[=*Value*]
Sets the specified extended attribute name to the specified *Value* for each file. If no *Value* is specified, `--set-attr` *AttributeName* sets the extended attribute name to a zero-length value.

--pure-attr-create
When this option is used, the command fails if the specified extended attribute exists.

--pure-attr-replace
When this option is used, the command fails if the specified extended attribute does not exist.

--delete-attr *AttributeName*
Removes the extended attribute.

For example, to remove wad, wadfg, and bgf, enter the following command:

```
mmchattr --delete-attr gpfs.WAD,gpfs.WADFG,gpfs.BGF
```

--pure-attr-delete
When this option is used, the command fails if the specified extended attribute does not exist.

--hex-attr
Inputs the attribute value in hex.

--hex-attr-name
Inputs the attribute name in hex.

--no-attr-ctime
Changes the attribute without setting the ctime of the file. This is restricted to root only.

[--compact[=*Option][,Option]...]]*
Where Option can be either *NumDirectoryEntries*, indblk or fragment.

Performs any of the three operations:

- Sets the minimum compaction size of the directories that are specified in the **Filename** parameter.
- Deletes the indirect blocks that are redundant.
- Reduces the last logical data block to the number of subblocks that are required to store the data.

NumDirectoryEntries

The minimum compaction size is the number of directory slots, including both full and empty slots, that a directory is allowed to retain when it is automatically compacted. By default, in IBM Spectrum Scale 4.1 or later, a directory is compacted as much as possible. However, in systems in which many files are added to and removed from a directory in a short time, file system performance might be improved by setting the minimum compaction size of the directory.

The compact parameter sets the minimum compaction size of a directory to the specified number of slots. For example, if a directory contains 5,000 files and you set the minimum compaction size to 50,000, then the file system adds 45,000 directory slots. The directory can grow beyond 50,000 entries, but the file system does not allow the directory to be compacted below 50,000 slots.

Set *NumDirectoryEntries* to the total number of directory slots that you want to keep, including files that the directory already contains. You can specify the number of directory slots either as an integer or as an integer followed by the letter k (1000 slots) or m (1,000,000 slots). If you expect the average length of file names to be greater than 19 bytes, calculate the number of slots by the following formula:

```
NumDirectoryEntries = n * ( 1 + ceiling((namelen - 19)/32)
```

where:

n
Specifies the number of entries (file names) in the directory.

ceiling()
A function that rounds a fractional number up to the next highest integer. For example, ceiling(1.03125) returns 2.

namelen
Specifies the expected average length of file names.

For example, if you want 50,000 entries with an average file name length of 48, then NumDirectoryEntries = 50000 * (1 + 1) = 100000.

To restore the default behavior of the file system, specify the compact=0. The directory is compacted as far as possible.

To see the current value of this parameter, run the mmlsattr command with the -L option. For more information, see the topic “mmlsattr command” on page 487. To set or read the value

of this parameter in a program, see the topics "gpfs_prealloc() subroutine" on page 958 and "gpfs_fstat_x() subroutine" on page 876.

Note: The *NumDirectoryEntries* value is not supported if the file system was created in IBM Spectrum Scale 4.1 or earlier. The **compact** parameter that is specified converts the directory to 4.1 format and compacts the directory as far as possible.

The **mmchattr --compact=NumDirectoryEntries** command run for a regular file will fail with the following error message `Compact failed: NumDirectoryEntries is supported only for directory`.

indblk
Deallocates all redundant indirect blocks of a regular file.

When a file is truncated to zero file size, its indirect blocks can be deallocated. However, in versions earlier than 5.0.4.1, IBM Spectrum Scale retains the redundant indirect blocks in such scenarios. For a file system that has a large number of files, the metadata disk space wasted by these indirect blocks can be significant. You can use the `indblk` option to deallocate the unnecessary indirect blocks.

fragment
Applies a ShrinktoFit operation on a regular file to reduce disk usage.

The `fragment` option reduces the last logical block of data of the file to the actual number of subblocks that are required. When a file is closed after being written to, GPFS tries to reduce the last logical block of data of the file to the actual number of subblocks required to save disk space. Under normal circumstances, the file is compacted when it is closed. In a few rare cases, however, the shrink might fail and leave a full data block for the last logical block of data of the file. The `fragment` option shrinks the last logical block to the actual number of subblocks required.

Note: Specify only the `--compact` parameter, without any of the suboptions, to perform both the `indblk` and `fragment` operations.

--compression {yes | no | z | lz4 | zfast | alphae | alphah}
Compresses or decompresses the specified files. The compression libraries are intended primarily for the following uses:

z
Cold data. Favors compression efficiency over access speed.

lz4
Active, nonspecific data. Favors access speed over compression efficiency.

zfast
Active genomic data in FASTA, SAM, or VCF format.

alphae
Active genomic data in FASTQ format. Slightly favors compression efficiency over access speed.

alphah
Active genomic data in FASTQ format. Slightly favors access speed over compression efficiency.

The following table summarizes the effect of each option on compressed or uncompressed files:

Table 13. Effects of options on compressed or uncompressed files.

Option	Uncompressed files	Compressed files that were generated with a different compression library	Compressed files that were generated with the same compression library	Will it benefit from the hardware compression feature that is available starting with IBM z15™?
yes	Compress with z	Not affected	Not affected	Yes
no	Not affected	Decompress	Decompress	Yes

Table 13. Effects of options on compressed or uncompressed files. (continued)

Option	Uncompressed files	Compressed files that were generated with a different compression library	Compressed files that were generated with the same compression library	Will it benefit from the hardware compression feature that is available starting with IBM z15™?
z	Compress with z	Re-compress with z	Not affected	Yes
lz4	Compress with lz4	Re-compress with lz4	Not affected	No
zfast	Compress with zfast	Re-compress with zfast	Not affected	Yes
alphae	Compress with alphae	Re-compress with alphae	Not affected	Yes
alphah	Compress with alphah	Re-compress with alphah	Not affected	Yes

You can use the `-I defer` option to defer the operation until a later call to `mmrestripefs` or `mmrestripefile`. For more information, see the topic *File compression* in the *IBM Spectrum Scale: Administration Guide*.

--block-group-factor *BlockGroupFactor*
Specifies how many file system blocks are laid out sequentially on disk to behave like a single large block. This option only works if `--allow-write-affinity` is set for the data pool. This applies only to a new data block layout; it does not migrate previously existing data blocks.

--write-affinity-depth *WriteAffinityDepth*
Specifies the allocation policy to be used. This option only works if `--allow-write-affinity` is set for the data pool. This applies only to a new data block layout; it does not migrate previously existing data blocks.

--write-affinity-failure-group "*WadfgValueString*"
Indicates the range of nodes (in a shared nothing architecture) where replicas of blocks in the file are to be written. You use this parameter to determine the layout of a file in the cluster so as to optimize the typical access patterns of your applications. This applies only to a new data block layout; it does not migrate previously existing data blocks.

"*WadfgValueString*" is a semicolon-separated string identifying one or more failure groups in the following format:

```
FailureGroup1[;FailureGroup2[;FailureGroup3]]
```

where each *FailureGroupx* is a comma-separated string identifying the rack (or range of racks), location (or range of locations), and node (or range of nodes) of the failure group in the following format:

```
Rack1{:Rack2{:...{:Rackx}}},Location1{:Location2{:...{:Locationx}}},ExtLg1{:ExtLg2{:...
{:ExtLgx}}}
```

For example, the following value

```
1,1,1:2;2,1,1:2;2,0,3:4
```

means that the first failure group is on rack 1, location 1, extLg 1 or 2; the second failure group is on rack 2, location 1, extLg 1 or 2; and the third failure group is on rack 2, location 0, extLg 3 or 4.

If the end part of a failure group string is missing, it is interpreted as 0. For example, the following are interpreted the same way:

```
2
2,0
2,0,0
```

Notes:

1. Only the end part of a failure group string can be left off. The missing end part may be the third field only, or it may be both the second and third fields; however, if the third field is provided, the second field must also be provided. The first field must *always* be provided. In other words, every comma must both follow and precede a number; therefore, *none* of the following are valid:

   ```
   2,0,
   2,
   ,0,0
   0,,0
   ,,0
   ```

2. Wildcard characters (*) are supported in these fields.

--indefinite-retention {yes | no}
Turns indefinite retention on or off. An alternative form of this parameter is `-e {yes | no}`. See `--expiration-time`.

--expiration-time *yyyy-mm-dd*[*@hh:mm:ss*]
Specifies the expiration time. An alternative form of this parameter is `-E` *yyyy-mm-dd*[*@hh:mm:ss*]. Expiration time and indefinite retention are independent attributes. You can change the value of either one without affecting the value of the other.

--inode-number [*SnapPath*/]*InodeNumber*
The inode number of the file to be changed. You must enter at least one inode number or file name, but not both; if you specify more than one inode number, delimit each inode number by a space. If the current working directory is not already inside the active file system or snapshot, then the inode number has to be prefixed by the path to the active file system or snapshot. For example:

```
mmchattr –inode-number /fs0/.snapshots/snap1/34608
```

You must have root authority to use this option.

Filename
The name of the file to be changed. You must enter at least one file name or inode number, but not both; if you specify more than one file name, delimit each file name by a space. Wildcard characters are supported in file names; for example, `project*.sched`.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have write access to the file to run the `mmchattr` command unless you are changing the `gpfs.readReplicaRule` extended attribute and you are the file owner, in which case read permission is sufficient.

You can issue the `mmchattr` command only from a node in the GPFS cluster where the file system is mounted.

Examples

1. To change the metadata replication factor to 2 and the data replication factor to 2 for the `project7.resource` file in file system `fs1`, issue the following command:

```
# mmchattr -m 2 -r 2 /fs1/project7.resource
```

   To confirm the change, issue the following command:

```
# mmlsattr project7.resource
```

   A sample output is as follows:

```
   replication factors
metadata(max) data(max) file    [flags]
------------- --------- --------------
       2 (  2)   2 (  2) /fs1/project7.resource
```

2. Migrating data from one storage pool to another using the `mmchattr` command with the `-I defer` option, or the `mmapplypolicy` command with the `-I defer` option causes the data to be ill-placed. This means that the storage pool assignment for the file has changed, but the file data has not yet been migrated to the assigned storage pool.

   The `mmlsattr -L` command causes show ill-placed flags on the files that are ill-placed. The `mmrestripefs`, or `mmrestripefile` command can be used to migrate data to the correct storage pool, and the ill-placed flag is cleared. This is an example of an ill-placed file:

```
# mmlsattr -L 16Kfile6.tmp
```

   A sample output is as follows:

```
file name:            16Kfile6.tmp
metadata replication: 1 max 2
data replication:     1 max 2
immutable:            no
appendOnly:           no
flags:                directio
storage pool name:    system
fileset name:         root
snapshot name:
creation time:        Thu Mar 28 14:49:23 2013
Misc attributes:      ARCHIVE
```

3. The following example shows the result of using the `--set-attr` parameter.

```
# mmchattr --set-attr user.pfs001=testuser 16Kfile7.tmp
# mmlsattr -L -d 16Kfile7.tmp
```

   A sample output is as follows:

```
file name:            16Kfile7.tmp
metadata replication: 1 max 2
data replication:     1 max 2
immutable:            no
appendOnly:           no
flags:
storage pool name:    system
fileset name:         root
snapshot name:
creation Time:        Fri Feb 24 12:00:13 2012
Misc attributes:      ARCHIVE
user.pfs001:         "testuser"
```

4. To set the write affinity failure group for a file and to see the results, issue the following commands:

```
# mmchattr --write-affinity-failure-group="64,0,0;128,0,1;128,0,2" /gpfs1/testfile
# mmlsattr -L /gpfs1/testfile
```

A sample output is as follows:

```
file name:            /gpfs1/testfile
metadata replication: 3 max 3
data replication:     3 max 3
immutable:            no
appendOnly:           no
flags:
storage pool name:    system
fileset name:         root
snapshot name:
Write Affinity Depth Failure Group(FG) Map for copy:1 64,0,0
Write Affinity Depth Failure Group(FG) Map for copy:2 128,0,1
Write Affinity Depth Failure Group(FG) Map for copy:3 128,0,2
creation time:        Wed Sep 12 02:53:18 2012
Misc attributes:      ARCHIVE
```

See also

- “mmcrfs command” on page 318
- “mmlsattr command” on page 487
- “mmlsfs command” on page 506

Location

/usr/lpp/mmfs/bin

mmchcluster command

Changes GPFS cluster configuration data.

Synopsis

```
mmchcluster --ccr-enable
```

or

```
mmchcluster {[--ccr-disable [--force]  [--force-ccr-disable-msg]]  [-p PrimaryServer] [-s
SecondaryServer]}
```

or

```
mmchcluster -p LATEST
```

or

```
mmchcluster {[-r RemoteShellCommand] [-R RemoteFileCopyCommand]
   [--nouse-sudo-wrapper]} | --use-sudo-wrapper
   [--sudo-user UserName]
```

or

```
mmchcluster -C ClusterName
```

Note: The primary and secondary configuration server functionality is deprecated and will be removed in a future release. The default configuration service is CCR. For now, you can switch from the CCR configuration service to the primary and secondary configuration servers by issuing the `mmchcluster` command with the `--ccr-disable` parameter. See the description of that parameter later in this topic. For more information, see the topic “mmcrcluster command” on page 306.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmchcluster` command serves several purposes. You can use it to do the following:

- Change the remote shell and remote file copy programs to be used by the nodes in the cluster.
- Change the cluster name.
- Enable or disable the cluster configuration repository (CCR).

When using the traditional server-based (non-CCR) configuration repository, you can also do the following:

- Change the primary or secondary GPFS cluster configuration server.
- Synchronize the primary GPFS cluster configuration server.

To display current system information for the cluster, issue the `mmlscluster` command.

For information on how to specify node names, see the topic *Specifying nodes as inputs to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

When issuing the `mmchcluster` command with the `-p` or `-s` options, the specified nodes must be available in order for the command to succeed. If any of the nodes listed are not available when the command is issued, a message listing those nodes is displayed. You must correct the problem on each node and reissue the command.

Attention: The `mmchcluster` command, when issued with either the `-p` or `-s` option, is designed to operate in an environment where the current primary and secondary cluster configuration servers are **not** available. As a result, the command can run without obtaining its regular serialization locks. To assure smooth transition to a new cluster configuration server, no other GPFS commands (`mm` commands) should be running when the command is issued, nor should any other command be issued until the `mmchcluster` command has successfully completed.

Parameters

--ccr-enable

Enables the configuration server repository (CCR), which stores redundant copies of configuration data files on all quorum nodes. The advantage of CCR over the traditional primary or backup configuration server semantics is that when using CCR, all GPFS administration commands as well as file system mounts and daemon startups work normally as long as a majority of quorum nodes are accessible.

For more information, see the topic *Cluster configuration data files* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

The CCR operation requires the use of the GSKit toolkit for authenticating network connections. As such, the `gpfs.gskit` package, which is available on all Editions, should be installed.

--ccr-disable [--force] [--force-ccr-disable-msg]

Closes the CCR environment and reverts the cluster to the primary and secondary configuration servers.

Attention: The primary and secondary configuration server functionality is deprecated and will be removed in a future release.

Before you change from the CCR environment, consider the following issues:

- When the CCR environment is closed, the monitoring function of the `mmhealth` command is automatically disabled.
- Certain services that are dependent on the CCR environment must be disabled before you can change from the CCR environment. These include Transparent Cloud Tiering, Watch Folder and Audit Logging, Call Home, CES, and the IBM Spectrum Scale GUI.
- You must shut down GPFS on all the nodes of the cluster before you close the CCR and restart GPFS afterward.

By default the `--ccr-disable` parameter causes the `mmchcluster` command to take the following precautionary actions before it closes the CCR:

- It displays a warning message and prompts you to confirm or cancel the decision to close the CCR. The warning message states that the primary and secondary configuration feature is deprecated and will be removed; that the monitoring function of the mmhealth command will be disabled; and that CCR-dependent services might be running. However, you can block this message and the confirmation prompt by specifying the `--force-ccr-disable-msg` parameter.
- It checks whether any services that are dependent on the CCR are running. If so the command displays an error message and ends without closing the CCR so that you can stop the services properly. You can block this check by specifying the `--force` parameter. The following services are dependent on CCR:
  - Transparent Cloud Tiering
  - Watch Folder and Audit Logging.
  - Call Home
  - The IBM Spectrum Scale GUI interface.
  - CES
    - To close CES, remove all the CES nodes from the cluster.

CAUTION: The `--ccr-disable` parameter causes the current CES configuration information to be deleted. If you have created CES configuration information, ensure that you have made a backup copy before you issue this command. For more information, see the topic *Backing up and restoring protocols and CES configuration information* in the *IBM Spectrum Scale: Administration Guide*.

You can specify either or both of the blocking parameters `--force` and `--force-ccr-disable-msg`.

--force
Causes the command to skip the check for services that are dependent on CCR that might be running. For more information, see the previous description of the `--ccr-disable` parameter.

Warning: If such services are running when the CCR is closed, they can be left in an undefined state and might generate errors in system log files.

--force-ccr-disable-msg
Causes the command to skip the general warning message and the prompt to cancel or confirm the closing of the CCR. For more information, see the previous description of the `--ccr-disable` parameter.

-p *PrimaryServer*
Change the primary server node for the GPFS cluster data. This may be specified as a short or long node name, an IP address, or a node number.

LATEST – Synchronize all of the nodes in the GPFS cluster ensuring they are using the most recently specified primary GPFS cluster configuration server. If an invocation of the `mmchcluster` command fails, you are prompted to reissue the command and specify LATEST on the `-p` option to synchronize all of the nodes in the GPFS cluster. Synchronization provides for all nodes in the GPFS cluster to use the most recently specified primary GPFS cluster configuration server.

This option only applies when the traditional server-based configuration (non-CCR) repository is used.

-s *SecondaryServer*
Change the secondary server node for the GPFS cluster data. To remove the secondary GPFS server and continue operating without it, specify a null string, `" "`, as the parameter. This may be specified as a short or long node name, an IP address, or a node number.

This option only applies when the traditional server-based configuration (non-CCR) repository is used.

-r *RemoteShellCommand*
Specifies the fully-qualified path name for the remote shell program to be used by GPFS.

The remote shell command must adhere to the same syntax format as the `ssh` command, but may implement an alternate authentication mechanism.

-R *RemoteFileCopy*
Specifies the fully-qualified path name for the remote file copy program to be used by GPFS.

The remote copy command must adhere to the same syntax format as the `scp` command, but may implement an alternate authentication mechanism.

--nouse-sudo-wrapper
Specifies that the cluster reverts to using the default remote shell program and remote copy program. For more information, see the topic *Running IBM Spectrum Scale without remote root login* in the *IBM Spectrum Scale: Administration Guide*.

`--use-sudo-wrapper` [`sudo-user` *UserName*]
Causes the nodes in the cluster to call the ssh and scp sudo wrapper scripts as the remote shell program and the remote copy program. For more information, see the topic *Running IBM Spectrum Scale without remote root login* in the *IBM Spectrum Scale: Administration Guide*.

--sudo-user *UserName*
Specifies a non-root admin user ID to be used when sudo wrappers are enabled and a root-level background process calls an administration command directly instead of through **sudo**. The GPFS

daemon that processes the administration command specifies this non-root user ID instead of the root ID when it needs to run internal commands on other nodes. For more information, see the topic *Root-level processes that call administration commands directly* in the *IBM Spectrum Scale: Administration Guide*.

To disable this feature, specify the key word **DELETE** instead of a user name, as in the following example:

```
# mmchcluster --sudo-user DELETE
```

-C *ClusterName*
Specifies a new name for the cluster. If the user-provided name contains dots then the command assumes that the user-provided name is a fully qualified domain name. Otherwise, to make the cluster name unique, the command appends the domain of a quorum node to the user-provided name. The maximum length of the cluster name including any appended domain name is 115 characters.

Since each cluster is managed independently, there is no automatic coordination and propagation of changes between clusters like there is between the nodes within a cluster. This means that if you change the name of the cluster, you should notify the administrators of all other GPFS clusters that can mount your file systems so that they can update their own environments.

Before running this option, ensure that all GPFS daemons on all nodes have been stopped.

See the `mmauth`, `mmremotecluster`, and `mmremotefs` commands.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmchcluster` command.

The node on which the command is issued must be able to run remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see the topic *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

Examples

To change the primary GPFS server for the cluster, issue the following command:

```
# mmchcluster -p k164n06
```

A sample output is as follows:

```
# mmchcluster: Command successfully completed
```

To confirm the change, issue this command:

```
# mmlscluster
```

A sample output is as follows:

```
GPFS cluster information
========================
  GPFS cluster name:       cluster1.kgn.ibm.com
  GPFS cluster id:         680681562214606028
  GPFS UID domain:         cluster1.kgn.ibm.com
  Remote shell command:      /usr/bin/rsh
  Remote file copy command:  /usr/bin/rcp
```

```
GPFS cluster configuration servers:
-----------------------------------
  Primary server:    k164n06.kgn.ibm.com
  Secondary server:  k164n05.kgn.ibm.com

Node Daemon node name    IP address   Admin node name   Designation
-----------------------------------------------------------------------
 1  k164n04.kgn.ibm.com  198.117.68.68  k164n04.kgn.ibm.com  quorum
 2  k164n05.kgn.ibm.com  198.117.68.71  k164n05.kgn.ibm.com  quorum
 3  k164n06.kgn.ibm.com  198.117.68.70  k164sn06.kgn.ibm.com
```

See also

- “mmaddnode command” on page 34
- “mmchnode command” on page 244
- “mmcrcluster command” on page 306
- “mmdelnode command” on page 375
- “mmlscluster command” on page 492
- “mmremotecluster command” on page 660

Location

/usr/lpp/mmfs/bin

mmchconfig command

Changes GPFS configuration parameters.

Synopsis

```
mmchconfig Attribute=value[,Attribute=value...] [-i | -I]
             [-N {Node[,Node...] | NodeFile | NodeClass}]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmchconfig` command to change the GPFS configuration attributes on a single node, a set of nodes, or globally for the entire cluster.

Results

The configuration is updated on the specified nodes.

Parameters

-I

Specifies that the changes take effect immediately, but do not persist when GPFS is restarted. This option is valid only for the following attributes:

- `deadlockBreakupDelay`
- `deadlockDataCollectionDailyLimit`
- `deadlockDataCollectionMinInterval`
- `deadlockDetectionThreshold`
- `deadlockDetectionThresholdForShortWaiters`
- `deadlockOverloadThreshold`
- `dioSmallSeqWriteBatching`
- `diskReadExclusionList`
- `dmapiMountEvent`
- `dmapiMountTimeout`
- `dmapiSessionFailureTimeout`
- `expelDataCollectionDailyLimit`
- `expelDataCollectionMinInterval`
- `fastestPolicyCmpThreshold`
- `fastestPolicyMaxValidPeriod`
- `fastestPolicyMinDiffPercent`
- `fastestPolicyNumReadSamples`
- `fileHeatLossPercent`
- `fileHeatPeriodMinutes`
- `ignorePrefetchLUNCount`
- `ignoreReplicationForQuota`
- `ignoreReplicationOnStatfs`

- linuxStatfsUnits
- lrocData
- lrocDataMaxFileSize
- lrocDataStubFileSize
- lrocDirectories
- lrocEnableStoringClearText
- lrocInodes
- logRecoveryThreadsPerLog
- logOpenParallelism
- logRecoveryParallelism
- maxMBpS
- nfsPrefetchStrategy
- nsdBufSpace
- nsdCksumTraditional
- nsdDumpBuffersOnCksumError
- nsdInlineWriteMax
- nsdMultiQueue
- pagepool
- panicOnIOHang
- pitWorkerThreadsPerNode
- proactiveReconnect
- readReplicaPolicy
- readReplicaRuleEnabled
- seqDiscardThreshold
- syncbuffsperiteration
- systemLogLevel
- unmountOnDiskFail
- verbsRdmaRoCEToS
- worker1Threads (only when adjusting value down)
- writebehindThreshold

-i

Specifies that the changes take effect immediately and are permanent. This option is valid only for the following attributes:

- cesSharedRoot
- cnfsGrace
- cnfsMountdPort
- cnfsNFSDprocs
- cnfsReboot
- cnfsSharedRoot
- cnfsVersions
- commandAudit
- confirmShutdownIfHarmful
- dataDiskWaitTimeForRecovery
- dataStructureDump

- deadlockBreakupDelay
- deadlockDataCollectionDailyLimit
- deadlockDataCollectionMinInterval
- deadlockDetectionThreshold
- deadlockDetectionThresholdForShortWaiters
- deadlockOverloadThreshold
- debugDataControl
- dioSmallSeqWriteBatching
- disableInodeUpdateOnFDatasync
- diskReadExclusionList
- dmapiMountEvent
- dmapiMountTimeout
- dmapiSessionFailureTimeout
- expelDataCollectionDailyLimit
- expelDataCollectionMinInterval
- fastestPolicyCmpThreshold
- fastestPolicyMaxValidPeriod
- fastestPolicyMinDiffPercent
- fastestPolicyNumReadSamples
- fileHeatLossPercent
- fileHeatPeriodMinutes
- ignorePrefetchLUNCount
- ignoreReplicationForQuota
- ignoreReplicationOnStatfs
- linuxStatfsUnits
- lrocData
- lrocDataMaxFileSize
- lrocDataStubFileSize
- lrocDirectories
- lrocEnableStoringClearText
- lrocInodes
- logRecoveryThreadsPerLog
- logOpenParallelism
- logRecoveryParallelism
- maxDownDisksForRecovery
- maxFailedNodesForRecovery
- maxMBpS
- metadataDiskWaitTimeForRecovery
- minDiskWaitTimeForRecovery
- mmfsLogTimeStampISO8601
- nfsPrefetchStrategy
- nsdBufSpace
- nsdCksumTraditional

- nsdDumpBuffersOnCksumError
- nsdInlineWriteMax
- nsdMultiQueue
- pagepool
- panicOnIOHang
- pitWorkerThreadsPerNode
- proactiveReconnect
- readReplicaPolicy
- readReplicaRuleEnabled
- restripeOnDiskFailure
- sdrNotifyAuthEnabled
- seqDiscardThreshold
- sudoUser
- syncbuffsperiteration
- systemLogLevel
- unmountOnDiskFail
- verbsRdmaRoCEToS
- worker1Threads (only when adjusting value down)
- writebehindThreshold

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies the set of nodes to which the configuration changes apply. The default is -N all.

For information on how to specify node names, see the topic *Specifying nodes as inputs to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

To see a complete list of the attributes for which the -N flag is valid, see the table "Configuration attributes on the mmchconfig command" in the topic *Changing the GPFS cluster configuration data* in the *IBM Spectrum Scale: Administration Guide*.

This command does not support a *NodeClass* of mount.

Attribute=value
Specifies the name of the attribute to be changed and its associated *value*. More than one attribute and value pair can be specified. To restore the GPFS default setting for an attribute, specify DEFAULT as its *value*.

This command accepts the following attributes:

adminMode
Specifies whether all nodes in the cluster are used for issuing GPFS administration commands or just a subset of the nodes. Valid values are:

allToAll
Indicates that all nodes in the cluster are used for running GPFS administration commands and that all nodes are able to execute remote commands on any other node in the cluster without the need of a password.

central
Indicates that only a subset of the nodes is used for running GPFS commands and that only those nodes are able to execute remote commands on the rest of the nodes in the cluster without the need of a password.

For more information, see the topic *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

afmAsyncDelay
Specifies (in seconds) the amount of time by which write operations are delayed (because write operations are asynchronous with respect to remote clusters). For write-intensive applications that keep writing to the same set of files, this delay is helpful because it replaces multiple writes to the home cluster with a single write containing the latest data. However, setting a very high value weakens the consistency of data on the remote cluster.

This configuration parameter is applicable only for writer caches (SW, IW, and primary), where data from cache is pushed to home.

Valid values are between 1 and 2147483647. The default is 15.

afmAsyncOpWaitTimeout
Specifies the time (in seconds) that AFM or AFM DR waits for completion of any inflight asynchronous operation which is synchronizing with the home or primary cluster. Subsequently, AFM or AFM DR cancels the operation and tries synchronization again after home or primary cluster is available.

Default value is 300. The range of valid values is 5 and 2147483647.

afmDirLookupRefreshInterval
Controls the frequency of data revalidations that are triggered by such lookup operations as `ls` or `stat` (specified in seconds). When a lookup operation is done on a directory, if the specified amount of time has passed, AFM sends a message to the home cluster to find out whether the metadata of that directory has been modified since the last time it was checked. If the time interval has not passed, AFM does not check the home cluster for updates to the metadata.

Valid values are 0 through 2147483647. The default is 60. In situations where home cluster data changes frequently, a value of 0 is recommended.

afmDirOpenRefreshInterval
Controls the frequency of data revalidations that are triggered by such I/O operations as `read` or `write` (specified in seconds). After a directory has been cached, open requests resulting from I/O operations on that object are directed to the cached directory until the specified amount of time has passed. Once the specified amount of time has passed, the open request gets directed to a gateway node rather than to the cached directory.

Valid values are between 0 and 2147483647. The default is 60. Setting a lower value guarantees a higher level of consistency.

afmDisconnectTimeout
The Waiting period in seconds to detect the status of the home cluster. If the home cluster is inaccessible, the metadata server (MDS) changes the state to 'disconnected'.

afmEnableNFSSec
If enabled at cache/primary, exported paths from home/secondary with kerberos-enabled security levels like `sys`, `krb5`, `krb5i`, `krb5p` are mounted at cache/primary in the increasing order of security level - `sys`, `krb5`, `krb5i`, `krb5p`. For example, the security level of exported path is `krb5i` then at cache, AFM/AFM DR tries to mount with level `sys`, followed by `krb5`, and finally mounts with the security level `krb5i`. If disabled at cache/primary, then exported paths from home or secondary are mounted with security level `sys` at cache or primary. You must configure KDC clients on all the gateway nodes at cache or primary before enabling this parameter.

Valid values are `yes` and `no`. The default value is `no`.

afmExpirationTimeout
Is used with `afmDisconnectTimeout` (which can be set only through `mmchconfig`) to control how long a network outage between the cache and home clusters can continue before the data in the cache is considered out of sync with home. After `afmDisconnectTimeout` expires, cached data remains available until `afmExpirationTimeout` expires, at which point the cached data is considered expired and cannot be read until a reconnect occurs.

Valid values are 0 through 2147483647. The default is disable.

afmFileLookupRefreshInterval
Controls the frequency of data revalidations that are triggered by such lookup operations as `ls` or `stat` (specified in seconds). When a lookup operation is done on a file, if the specified amount of time has passed, AFM sends a message to the home cluster to find out whether the metadata of the file has been modified since the last time it was checked. If the time interval has not passed, AFM does not check the home cluster for updates to the metadata.

Valid values are 0 through 2147483647. The default is 30. In situations where home cluster data changes frequently, a value of 0 is recommended.

afmFileOpenRefreshInterval
Controls the frequency of data revalidations that are triggered by such I/O operations as `read` or `write` (specified in seconds). After a file has been cached, open requests resulting from I/O operations on that object are directed to the cached file until the specified amount of time has passed. Once the specified amount of time has passed, the open request gets directed to a gateway node rather than to the cached file.

Valid values are 0 through 2147483647. The default is 30. Setting a lower value guarantees a higher level of consistency.

afmHardMemThreshold

Sets a limit to the maximum amount of memory that AFM can use on each gateway node to record changes to the file system. After this limit is reached, the fileset goes into a 'dropped' state.

Exceeding the limit and the fileset going into a 'dropped' state due to accumulated pending requests might occur if -

- the cache cluster is disconnected for an extended period of time.
- the connection with the home cluster is on a low bandwidth.

afmHashVersion

Specifies the version of hashing algorithm to be used to assign AFM and AFM ADR filesets across gateway nodes, thus running as few recoveries as possible. This minimizes impact of gateway nodes joining or leaving the active cluster.

Valid values are 1,2, 4 or 5. Default value is 2.

afmMaxParallelRecoveries
Specifies the number of filesets per gateway node on which event recovery is run. The default value is 0. When the value is 0, event recovery is run on all filesets of the gateway node.

afmNumReadThreads
Defines the number of threads that can be used on each participating gateway node during parallel read. The default value of this parameter is 1; that is, one reader thread will be active on every gateway node for each big read operation qualifying for splitting per the parallel read threshold value. The valid range of values is 1 to 64.

afmNumWriteThreads
Defines the number of threads that can be used on each participating gateway node during parallel write. The default value of this parameter is 1; that is, one writer thread will be active on every gateway node for each big write operation qualifying for splitting per the parallel write threshold value. Valid values can range from 1 to 64.

afmParallelMounts

When this parameter is enabled, the primary gateway node of a fileset at a cache cluster attempts to mount the exported path from multiple NFS servers that are defined in the mapping. Then, this primary gateway node sends unique messages through each NFS mount to improve performance by transferring data in parallel.

Before enabling this parameter, define the mapping between the primary gateway node and NFS servers by issuing the **mmafmconfig** command.

afmParallelReadChunkSize
Defines the minimum chunk size of the read that needs to be distributed among the gateway nodes during parallel reads. Values are interpreted in terms of bytes. The default value of this parameter is 128 MiB, and the valid range of values is 0 to 2147483647. It can be changed cluster wide with the `mmchconfig` command. It can be set at fileset level using `mmcrfileset` or `mmchfileset` commands.

afmParallelReadThreshold
Defines the threshold beyond which parallel reads become effective. Reads are split into chunks when file size exceeds this threshold value. Values are interpreted in terms of MiB. The default value is 1024 MiB. The valid range of values is 0 to 2147483647. It can be changed cluster wide with the `mmchconfig` command. It can be set at fileset level using `mmcrfileset` or `mmchfileset` commands.

afmParallelWriteChunkSize
Defines the minimum chunk size of the write that needs to be distributed among the gateway nodes during parallel writes. Values are interpreted in terms of bytes. The default value of this parameter is 128 MiB, and the valid range of values is 0 to 2147483647. It can be changed cluster wide with the `mmchconfig` command. It can be set at fileset level using `mmcrfileset` or `mmchfileset` commands.

afmParallelWriteThreshold
Defines the threshold beyond which parallel writes become effective. Writes are split into chunks when file size exceeds this threshold value. Values are interpreted in terms of MiB. The default value of this parameter is 1024 MiB, and the valid range of values is 0 to 2147483647. It can be changed cluster wide with the `mmchconfig` command. It can be set at fileset level using `mmcrfileset` or `mmchfileset` commands.

afmReadSparseThreshold
Specifies the size in MB for files in cache beyond which sparseness is maintained. For all files below the specified threshold, sparseness is not maintained.

afmRefreshAsync
Modifies the cache data refresh operation to asynchronous mode. Cache data refresh operation in asynchronous mode improves performance of applications that query data. Upon readdir or lookup request, a revalidation request for files or directories is queued as an asynchronous request to the gateway but the last known synchronized state of the cache data is returned to the applications. Cache data is refreshed after revalidation with home is complete. Revalidation time depends on the network availability and its bandwidth.

Valid values are 'no' and 'yes'. With the default value as 'no', cache data is validated with home synchronously. Specify the value as 'yes' if you want the cache data refresh operation to be in asynchronous mode.

afmRevalOpWaitTimeout
Specifies the time that AFM waits for revalidation to get response from the home cluster. Revalidation checks if any changes are available at home (data and metadata) that need to be updated to the cache cluster. Revalidation is performed when application trigger operations like lookup, open at cache. If revalidation is not completed within this time, AFM cancels the operation and returns data available at cache to the application.

Default value is 180. The range of valid values is 5 and 2147483647.

afmRPO
Specifies the recovery point objective (RPO) interval for an AFM DR fileset. This attribute is disabled by default. You can specify a value with the suffix `M` for minutes, `H` for hours, or `W` for weeks. For example, for 12 hours specify 12H. If you do not add a suffix, the value is assumed to be in minutes. The range of valid values is 720 minutes - 2147483647 minutes.

afmSecondaryRW
Specifies if the secondary is read-write or not.

yes
Specifies that the secondary is read-write.

no
Specifies that the secondary is not read-write.

afmShowHomeSnapshot
Controls the visibility of the home snapshot directory in cache. For this to be visible in cache, this variable has to be set to `yes`, and the snapshot directory name in the cache and home cannot be the same.

yes
Specifies that the home snapshot link directory is visible.

no
Specifies that the home snapshot link directory is not visible.

For more information about the snapshot, see *Peer snapshot -psnap* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

afmSyncOpWaitTimeout
Specifies the time that AFM or AFM DR waits for completion of any inflight synchronous operation which is synchronizing with the home or primary cluster. When any application is performing any synchronous operation at cache or secondary, AFM or AFM DR tries to get a response from home or primary cluster. If home or primary cluster is not responding, application might be unresponsive. If operation does not complete in this timeout interval, AFM or AFM DR cancels the operation.

Default value is 180. The range of valid values is 5 and 2147483647.

atimeDeferredSeconds
Controls the update behavior of `atime` when the `relatime` option is enabled. The default value is 86400 seconds (24 hours). A value of 0 effectively disables `relatime` and causes the behavior to be the same as the `atime` setting.

For more information, see the topic *Mount options specific to IBM Spectrum Scale* in the *IBM Spectrum Scale: Administration Guide*.

autoBuildGPL={yes | no | *mmbuildgplOptions*}
Causes IBM Spectrum Scale to detect when the GPFS portability layer (GPL) needs to be rebuilt and to rebuild it automatically. A rebuild is triggered if the GPFS kernel module is missing or if a new level of IBM Spectrum Scale is installed. The `mmbuildgpl` command is called to do the rebuild. For the rebuild to be successful, the requirements of the `mmbuildgpl` command must be met; in particular, the build tools and kernel headers must be present on each node. This attribute takes effect when the GPFS daemon is restarted. For more information, see the topics "mmbuildgpl command" on page 113 and *Using the mmbuildgpl command to build the GPFS portability layer on Linux nodes* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Note: This parameter does not apply to the AIX and Windows environments.

yes
Causes the GPL to be rebuilt when necessary.

no
Takes no action when the GPL needs to be rebuilt. This is the default value.

mmbuildgplOptions
Causes the GPL to be rebuilt when necessary and causes `mmbuildgpl` to be called with the indicated options. This value is a hyphen-separated list of options in any order:

quiet
Causes `mmbuildgpl` to be called with the **`--quiet`** parameter.

verbose
Causes `mmbuildgpl` to be called with the **`-v`** option.

Note that **yes**, **no**, and *mmbuildgplOptions* are mutually exclusive, and that *mmbuildgplOptions* implies **yes**. You cannot specify both **yes** and *mmbuildgplOptions*. You can specify both

quiet and **verbose** on the command line by separating them with a hyphen, as in `autoBuildGPL=quiet-verbose`. See Table 14 on page 178.

The `-N` flag is valid for this attribute.

Table 14. Values assigned to autoBuildGPL and their effects

Value assigned to autoBuildGPL	Command and options that are invoked when the GPL needs to be rebuilt
autoBuildGPL=no	N/A
autoBuildGPL=yes	mmbuildgpl
autoBuildGPL=quiet	mmbuildgpl --quiet
autoBuildGPL=verbose	mmbuildgpl -v
autoBuildGPL=quiet-verbose	mmbuildgpl --quiet -v
autoBuildGPL=verbose-quiet	mmbuildgpl --quiet -v

autoload
Starts GPFS automatically whenever the nodes are rebooted. Valid values are `yes` or `no`.

The `-N` flag is valid for this attribute.

automountDir
Specifies the directory to be used by the Linux automounter for GPFS file systems that are being mounted automatically. The default directory is `/gpfs/automountdir`. This parameter does not apply to AIX and Windows environments.

backgroundSpaceReclaimThreshold
Specifies the percentage of reclaimable blocks that must occur in an allocation space for devices capable of space reclaim, such as NVMe and thin provisioned disks, to trigger a background space reclaim. The default value is 0 indicating that background space reclaim is disabled. You can enable it by setting it to a value larger than 0 but less than or equal to 100. Specifying a lower value causes the background space reclaim to occur more frequently.

cesSharedRoot
Specifies a directory in a GPFS file system to be used by the Cluster Export Services (CES) subsystem. For the CES shared root, the recommended value is a dedicated file system, but it is not enforced. The CES shared root can also be a part of an existing GPFS file system. In any case, `cesSharedRoot` must reside on GPFS and must be available when it is configured through **mmchconfig**.

GPFS must be down on all CES nodes in the cluster when changing the `cesSharedRoot` attribute.

cifsBypassTraversalChecking
Controls the GPFS behavior while performing access checks for directories

GPFS grants the SEARCH access when the following conditions are met:

- The object is a directory
- The parameter value is **yes**
- The calling process is a Samba process

GPFS grants the SEARCH access regardless of the mode or ACL.

cipherList
Sets the security mode for the cluster. The security mode determines the level of the security that the cluster provides for communications between nodes in the cluster and also for communications with other clusters. There are three security modes:

EMPTY
The sending node and the receiving node do not authenticate each other, do not encrypt transmitted data, and do not check data integrity.

AUTHONLY
The sending and receiving nodes authenticate each other, but they do not encrypt transmitted data and do not check data integrity. This mode is the default in IBM Spectrum Scale 4.2 or later.

Cipher
The sending and receiving nodes authenticate each other, encrypt transmitted data, and check data integrity. To set this mode, you must specify the name of a supported cipher, such as AES128-GCM-SHA256.

Note: Although after **mmchconfig** is issued, the `mmfsd` daemon accepts the new `cipherList` immediately, it uses the cipher only for new TCP/TLS connections to other nodes. Existing `mmfsd` daemon connections remain with the prior `cipherList` settings. For the new `cipherList` to take complete effect immediately, GPFS needs to be restarted on all nodes in a rolling fashion, one node at a time, to prevent cluster outage. When a cipher other than *AUTHONLY* or *EMPTY* is in effect, it can lead to significant performance degradation, as this results in encryption and data integrity verification of the transmitted data.

For more information about the security mode and supported ciphers, see the topic *Security mode* in the *IBM Spectrum Scale: Administration Guide*.

cnfsGrace
Specifies the number of seconds a CNFS node will deny new client requests after a node failover or failback, to allow clients with existing locks to reclaim them without the possibility of some other client that is being granted a conflicting access. For v3, only new lock requests are denied. For v4, new lock, read, and write requests are rejected. The `cnfsGrace` value also determines the time period for the server lease.

Valid values are 10 - 600. The default is 90 seconds. While a short grace period is good for fast server failover, it comes at the cost of increased load on server to effect lease renewal.

GPFS must be down on all CNFS nodes in the cluster when changing the `cnfsGrace` attribute.

cnfsMountdPort
Specifies the port number to be used for **rpc.mountd**. See the *IBM Spectrum Scale: Administration Guide* for restrictions and additional information.

cnfsNFSDprocs
Specifies the number of **nfsd** kernel threads. The default is 32.

cnfsReboot
Specifies whether the node reboots when CNFS monitoring detects an unrecoverable problem that can be handled only by node failover.

Valid values are `yes` or `no`. The default is `yes` and recommended. If node reboot is not desired for other reasons, it should be noted that clients that were communicating with the failing node are likely to get errors or hang. CNFS failover is only guaranteed with `cnfsReboot` enabled.

The `-N` flag is valid for this attribute.

cnfsSharedRoot
Specifies a directory in a GPFS file system to be used by the clustered NFS subsystem.

GPFS must be down on all CNFS nodes in the cluster when changing the `cnfsSharedRoot` attribute.

See the *IBM Spectrum Scale: Administration Guide* for restrictions and additional information.

cnfsVersions
Specifies a comma-separated list of protocol versions that CNFS should start and monitor.

The default is 3,4.

GPFS must be down on all CNFS nodes in the cluster when changing the `cnfsVersions` attribute.

See the *IBM Spectrum Scale: Administration Guide* for additional information.

commandAudit
Controls the logging of audit messages for GPFS commands that change the configuration of the cluster. This attribute is not supported on Windows operating systems. For more information, see the topic *Audit messages for cluster configuration changes* in the *IBM Spectrum Scale: Problem Determination Guide*.

on
Starts audit messages. Messages go to syslog and the GPFS log.

syslogOnly
Starts audit messages. Messages go to syslog only. This value is the default.

off
Stops audit messages.

The `-N` flag is valid for this attribute.

confirmShutdownIfHarmful={yes|no}
Specifies whether the **mmshutdown** command checks that shutting down the listed nodes will cause a loss of function in the cluster. For more information, see "mmshutdown command" on page 706.

The default value is `yes`.

dataDiskCacheProtectionMethod
The `dataDiskCacheProtectionMethod` parameter defines the cache protection method for disks that are used for the GPFS file system. The valid values for this parameter are 0, 1, and 2.

The default value is 0. The default value indicates that the disks are Power-Protected and, when the down disk is started, only the standard GPFS log recovery is required. If the value of this parameter is 1, the disks are Power-Protected with no disk cache. GPFS works the same as before. If the value of this parameter is 2, when a node stops functioning, files that have data in disk cache must be recovered to a consistent state when the disk is started.

This parameter impacts only disks in the FPO storage pool. If the physical disk-write cache is enabled, the value of this parameter must be set to 2. Otherwise, maintain the default.

dataDiskWaitTimeForRecovery
Specifies a period, in seconds, during which the recovery of `dataOnly` disks is suspended to give the disk subsystem a chance to correct itself. This parameter is taken into account when the affected disks belong to a single failure group. If more than one failure group is affected, the delay is based on the value of `minDiskWaitTimeForRecovery`.

Valid values are 0 - 3600 seconds. The default is 3600. If `restripeOnDiskFailure` is no, `dataDiskWaitTimeForRecovery` has no effect.

dataStructureDump
Specifies a path for storing dumps. You can specify a directory or a symbolic link. The default is to store dumps in `/tmp/mmfs`. This attribute takes effect immediately whether or not `-i` is specified.

It is a good idea to create a directory or a symbolic link for problem determination information. Do not put it in a GPFS file system, because it might not be available if GPFS fails. When a problem occurs, GPFS can write 200 MiB or more of problem determination data into the directory. Copy and delete the files promptly so that you do not get a `NOSPACE` error if another failure occurs.

Important: Before you change the value of `dataStructureDump`, stop the GPFS trace. Otherwise you will lose GPFS trace data. Restart the GPFS trace afterward. For more information, see the topic *Generating GPFS trace reports* in the *IBM Spectrum Scale: Problem Determination Guide*.

The `-N` flag is valid for this attribute.

deadlockBreakupDelay
Specifies how long to wait after a deadlock is detected before attempting to break up the deadlock. Enough time must be provided to allow the debug data collection to complete.

The default is 0, which means that the automated deadlock breakup is disabled. A positive value enables the automated deadlock breakup. If automated deadlock breakup is to be enabled, a delay of 300 seconds or longer is recommended.

deadlockDataCollectionDailyLimit
Specifies the maximum number of times that debug data can be collected each day.

The default is 3. If the value is 0, then no debug data is collected when a potential deadlock is detected.

deadlockDataCollectionMinInterval
Specifies the minimum interval between two consecutive collections of debug data.

The default is 3600 seconds.

deadlockDetectionThreshold
Specifies the initial deadlock detection threshold. The effective deadlock detection threshold adjusts itself over time. A suspected deadlock is detected when a waiter waits longer than the effective deadlock detection threshold.

The default is 300 seconds. If the value is 0, then automated deadlock detection is disabled.

deadlockDetectionThresholdForShortWaiters
Specifies the deadlock detection threshold for short waiters. The default value is 60 seconds. Do not set a large value, because short waiters are supposed to complete and disappear quickly.

deadlockOverloadThreshold
Specifies the threshold for detecting a cluster overload condition. If the overload index on a node exceeds the deadlockOverloadThreshold, then the effective deadlockDetectionThreshold is raised. The overload index is calculated heuristically and is based mainly on the I/O completion times.

The default is 1. If the value is 0, then overload detection is disabled.

debugDataControl
Controls the amount of debug data that is collected. This attribute takes effect immediately whether or not `-i` is specified. The `-N` flag is valid for this attribute.

none
No debug data is collected.

`light`
The minimum amount of debug data that is most important for debugging issues is collected. This is the default value.

`medium`
More debug data is collected.

`heavy`
The maximum amount of debug data is collected, targeting internal test systems.

`verbose`
Needed only for troubleshooting special cases and can result in large dumps.

The following table provides more information about these settings:

Table 15. Settings for debugDataControl

Setting	Collect dump data	Collect trace information (if tracing is already running)	Collect a short sample of trace information[1]
none	No	No	No

Table 15. Settings for debugDataControl (continued)

Setting	Collect dump data	Collect trace information (if tracing is already running)	Collect a short sample of trace information[1]
light (default)	Yes	Yes	No
medium	Yes, more dump data	Yes	No
heavy (for internal test teams)	Yes, even more dump data	Yes	Yes
verbose (for developers)	Yes, all dump data	Yes	Yes

[1]If trace is not running, turn tracing on, let it run for 20 seconds, and then turn trace off.

defaultHelperNodes
Specifies a default set of nodes that can be used by commands that are able to distribute work to multiple nodes. To specify values for this parameter, follow the rules that are described for the `-N` option in the topic *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

To override this setting when you use such commands, explicitly specify the helper nodes with the `-N` option of the command that you issue.

The following commands can use the nodes that this parameter provides: `mmadddisk`, `mmapplypolicy`, `mmbackup`, `mmchdisk`, `mmcheckquota`, `mmdefragfs`, `mmdeldisk`, `mmdelsnapshot`, `mmfileid`, `mmfsck`, `mmimgbackup`, `mmimgrestore`, `mmrestorefs`, `mmrestripefs`, and `mmrpldisk`.

When the command runs, its lists the *NodeClass* values.

defaultMountDir
Specifies the default parent directory for GPFS file systems. The default value is `/gpfs`. If an explicit mount directory is not provided with the `mmcrfs`, `mmchfs`, or `mmremotefs` command, the default mount point is set to *DefaultMountDir*/*DeviceName*.

dioSmallSeqWriteBatching={yes | no}
Controls whether GPFS enables a performance optimization that allows multiple Direct I/O (DIO) Asynchronous Input/Output (AIO) write requests to be handled as buffered I/O and be batched together into larger write operations. When enabled, GPFS tries to combine multiple small sequential asynchronous Direct I/O writes when committing the writes to storage. Valid values are `yes` or no. The default value is no.

When `dioSmallSeqWriteBatching` is set to `yes` GPFS holds small (up to 64 KiB) AIO/DIO write requests for a few microseconds, to allow for the held request to be combined together with additional contiguous writes that might occur.

disableInodeUpdateOnFdatasync
Controls the inode update on fdatasync for mtime and atime updates. Valid values are `yes` or no.

When `disableInodeUpdateOnFdatasync` is set to `yes`, the inode object is not updated on disk for mtime and atime updates on `fdatasync()` calls. File size updates are always synced to the disk.

When `disableInodeUpdateOnFdatasync` is set to no, the inode object is updated with the current mtime on `fdatasync()` calls. This is the default.

diskReadExclusionList
Specifies the list of NSD names that should be excluded from data block reads. It is used for resolving data block replica mismatches. Separate the NSD names with a semicolon (;) and enclose the list in quotes. For example:

```
diskReadExclusionList="gpfs1nsd;gpfs2nsd;gpfs3nsd"
```

To disable this option, use:

```
diskReadExclusionList=""
```

You can also use:

```
diskReadExclusionList=DEFAULT
```

For more information, see the *Replica mismatches* topic in the *IBM Spectrum Scale: Problem Determination Guide*.

dmapiDataEventRetry
Controls how GPFS handles data events that are enabled again immediately after the event is handled by the DMAPI application. Valid values are as follows:

-1
Specifies that GPFS always regenerates the event as long as it is enabled. This value should be used only when the DMAPI application recalls and migrates the same file in parallel by many processes at the same time.

0
Specifies to never regenerate the event. Do not use this value if a file might be migrated and recalled at the same time.

RetryCount
Specifies the number of times the data event should be retried. The default is 2.

For further information regarding DMAPI for GPFS, see *GPFS-specific DMAPI events* in the *IBM Spectrum Scale: Command and Programming Reference*.

dmapiEventTimeout
Controls the blocking of file operation threads of NFS, while in the kernel waiting for the handling of a DMAPI synchronous event. The parameter value is the maximum time, in milliseconds, the thread blocks. When this time expires, the file operation returns ENOTREADY, and the event continues asynchronously. The NFS server is expected to repeatedly retry the operation, which eventually finds the response of the original event and continue. This mechanism applies only to read, write, and truncate event types, and only when such events come from NFS server threads. The timeout value is given in milliseconds. The value 0 indicates immediate timeout (fully asynchronous event). A value greater than or equal to 86400000 (which is 24 hours) is considered *infinity* (no timeout, fully synchronous event). The default value is 86400000.

For the parameter change to take effect, restart the GPFS daemon on the nodes that are specified in the `-N` option. If the `-N` option is not used, restart the GPFS daemon on all nodes.

For further information about DMAPI for GPFS, see *GPFS-specific DMAPI events* in the *IBM Spectrum Scale: Command and Programming Reference*.

The `-N` flag is valid for this attribute.

dmapiMountEvent
Controls the generation of the `mount`, `preunmount`, and `unmount` events. Valid values are:

all
`mount`, `preunmount`, and `unmount` events are generated on each node. This is the default behavior.

SessionNode
mount, preunmount, and unmount events are generated on each node and are delivered to the session node, but the session node does not deliver the event to the DMAPI application unless the event is originated from the SessionNode itself.

LocalNode
mount, preunmount, and unmount events are generated only if the node is a session node.

For further information regarding DMAPI for GPFS, see *GPFS-specific DMAPI events* in the *IBM Spectrum Scale: Command and Programming Reference*.

dmapiMountTimeout
Controls the blocking of mount operations, waiting for a disposition for the mount event to be set. This timeout is activated, at most once on each node, by the first external mount of a file system that has DMAPI enabled, and only if there has never before been a mount disposition. Any mount operation on this node that starts while the timeout period is active waits for the mount disposition. The parameter value is the maximum time, in seconds, that the mount operation waits for a disposition. When this time expires and there is still no disposition for the mount event, the mount operation fails, returning the EIO error. The timeout value is given in full seconds. The value 0 indicates immediate timeout (immediate failure of the mount operation). A value greater than or equal to 86400 (which is 24 hours) is considered *infinity* (no timeout, indefinite blocking until there is a disposition). The default value is 60.

The -N flag is valid for this attribute.

For further information regarding DMAPI for GPFS, see *GPFS-specific DMAPI events* in the *IBM Spectrum Scale: Command and Programming Reference*.

dmapiSessionFailureTimeout
Controls the blocking of file operation threads, while in the kernel, waiting for the handling of a DMAPI synchronous event that is enqueued on a session that has experienced a failure. The parameter value is the maximum time, in seconds, the thread waits for the recovery of the failed session. When this time expires and the session has not yet recovered, the event is canceled and the file operation fails, returning the EIO error. The timeout value is given in full seconds. The value 0 indicates immediate timeout (immediate failure of the file operation). A value greater than or equal to 86400 (which is 24 hours) is considered *infinity* (no timeout, indefinite blocking until the session recovers). The default value is 0.

For further information regarding DMAPI for GPFS, see *GPFS-specific DMAPI events* in the *IBM Spectrum Scale: Command and Programming Reference*.

The -N flag is valid for this attribute.

enableIPv6
Controls whether the GPFS daemons communicate through the IPv6 network. The following values are valid:

no
Specifies that the GPFS daemons do not communicate through the IPv6 network. This is the default value.

Note: If any of the node interfaces that are specified for the mmcrcluster command resolves to an IPv6 address, the mmcrcluster command automatically enables the new cluster for IPv6 and sets the enableIPv6 attribute to yes. For more information, see *Enabling a cluster for IPv6* in the *IBM Spectrum Scale: Administration Guide*.

yes
Specifies that the GPFS daemons communicate through the IPv6 network. yes requires that the daemon be down on all nodes.

prepare
After the command completes, the daemons can be recycled on all nodes at a time chosen by the user (before proceeding to run the command with commit specified).

commit
Verifies that all currently active daemons have received the new value, allowing the user to add IPv6 nodes to the cluster.

Note:

Before changing the value of **enableIPv6**, the GPFS daemon on the primary configuration server must be inactive. After changing the parameter, the GPFS daemon on the rest of nodes in the cluster should be recycled. This can be done one node a time.

To use IPv6 addresses for GPFS, the operating system must be properly configured as IPv6 enabled, and IPv6 addresses must be configured on all the nodes within the cluster.

encryptionKeyCacheExpiration
Specifies the refresh interval, in seconds, of the file system encryption key cache that is used internally by the `mmfsd` daemon. The default value of this parameter is 900 seconds. The refresh operation of the encryption key cache requires the remote key server to be accessible and functional. A restart of the GPFS `mmfsd` daemon is required for any change in the value of this parameter to become effective. For more information, see *Encryption keys* in the *IBM Spectrum Scale: Administration Guide*.

A value of 0 indicates that the encryption key cache does not expire and it is not periodically refreshed.

A value of 60 (seconds) or greater indicates that the encryption key cache expires after the specified amount of time. The encryption key cache is refreshed automatically by the `mmfsd` daemon from the remote key server. No administrative action is required.

Note:

Changing the value of the **encryptionKeyCacheExpiration** requires cluster services (`mmfsd`) to be restarted on each node in order to take effect.

Though a value of 0 means the encryption key cache does not expire and is not refreshed periodically, connectivity to the remote key server is required from all nodes that access an encrypted file system. Regardless of the value of the **encryptionKeyCacheExpiration**, access to the remote key server is required after the following scenarios:

- When you are unmounting or mounting an encrypted file system.
- When you are using the `mmchpolicy` command to install a new policy for the file system, even if the key does not change.
- When a KMIP client is registered or deregistered by using the `mmkeyserv client register` or `mmkeyserv client deregister` command.

enforceFilesetQuotaOnRoot
Controls whether fileset quotas should be enforced for the root user the same way as for any other users. Valid values are `yes` or `no`. The default is `no`.

expelDataCollectionDailyLimit
Specifies the maximum number of times that debug data associated with expelling nodes can be collected in a 24-hour period. Sometimes exceptions are made to help capture the most relevant debug data.

The default is 3. If the value is 0, then no expel-related debug data is collected.

expelDataCollectionMinInterval
Specifies the minimum interval, in seconds, between two consecutive expel-related data collection attempts on the same node.

The default is 3600 seconds.

failureDetectionTime
Indicates to GPFS the amount of time it takes to detect that a node has failed.

GPFS must be down on all the nodes when changing the `failureDetectionTime` attribute.

fastestPolicyCmpThreshold
Indicates the disk comparison count threshold, above which GPFS forces selection of this disk as the preferred disk to read and update its current speed.

Valid values are >= 3. The default is 50. In a system with SSD and regular disks, the value of the `fastestPolicyCmpThreshold` parameter can be set to a greater number to let GPFS refresh the speed statistics for slower disks less frequently.

fastestPolicyMaxValidPeriod
Indicates the time period after which the disk's current evaluation is considered invalid (even if its comparison count has exceeded the threshold) and GPFS prefers to read this disk in the next selection to update its latest speed evaluation.

Valid values are >= 1 in seconds. The default is 600 (10 minutes).

fastestPolicyMinDiffPercent
A percentage value indicating how GPFS selects the fastest between two disks. For example, if you use the default fastestPolicyMinDiffPercent value of 50, GPFS selects a disk as faster only if it is 50% faster than the other. Otherwise, the disks remain in the existing read order.

Valid values are 0 - 100 in percentage points. The default is 50.

fastestPolicyNumReadSamples
Controls how many read samples are taken to evaluate the disk's recent speed.

Valid values are 3 - 100. The default is 5.

fileHeatLossPercent
The file heat attribute of a file increases in value when the file is accessed but decreases in value over time if the file is not accessed. The **fileHeatLossPercent** attribute specifies the percent of file access heat that an unaccessed file loses at the end of each tracking period. The valid range is 0 - 100. The default value is 10, which indicates that an unaccessed file loses 10 percent of its file access heat at the end of each tracking period. The tracking period is set by **fileHeatPeriodMinutes**. For more information, see the topic *File heat: Tracking the file access temperature* in the *IBM Spectrum Scale: Command and Programming Reference*.

This attribute does not take effect until the GPFS daemon is stopped and restarted.

fileHeatPeriodMinutes
A nonzero value enables file heat tracking and specifies the frequency with which the file heat attribute is updated. A value of 0 disables file access heat tracking. The default value is 0. For more information, see the topic *File heat: Tracking the file access temperature* in the *IBM Spectrum Scale: Command and Programming Reference*.

This attribute does not take effect until the GPFS daemon is stopped and restarted.

FIPS1402mode
Controls whether GPFS uses a FIPS-140-2-compliant encryption module for encrypted communications between nodes and for file encryption. Valid values are `yes` or `no`. The default value is `no`.

When it is enabled, FIPS 140-2 mode applies only to the following two features of IBM Spectrum Scale:

- Encryption and decryption of file data when it is transmitted between nodes in the current cluster or between a node in the current cluster and a node in another cluster. To enable this feature, issue the following command:

  ```
  mmchconfig cipherList=SupportedCipher
  ```

  where *SupportedCipher* is a cipher that is supported by IBM Spectrum Scale, such as AES128-GCM-SHA256. For more information, see the following topics:

  – *Security mode* in the *IBM Spectrum Scale: Administration Guide*.

  - *Setting security mode for internode communications in a cluster* in the *IBM Spectrum Scale: Administration Guide*.

- Encryption of file data as it is written to storage media and decryption of file data as it is read from storage media. For more information about file data encryption, see the following section of the documentation:

  - *Encryption* in the *IBM Spectrum Scale: Administration Guide*.

  Note: For performance reasons, do not enable FIPS 140-2 mode unless all the nodes in the cluster are running FIPS-certified kernels in FIPS mode. This note applies only to encryption of file data as it is written to storage media and decryption of file data as it is read from storage media. This note does not apply to encryption and decryption of file data when it is transmitted between nodes.

FIPS 140-2 mode does not apply to other components of IBM Spectrum Scale that use encryption, such as object encryption.

frequentLeaveCountThreshold
Specifies the number of times a node exits the cluster within the last **frequentLeaveTimespanMinutes** before autorecovery ignores the next exit of that node. If the exit count of a node within the last **frequentLeaveTimespanMinutes** is greater than **frequentLeaveCountThreshold**, autorecovery ignores the corresponding node exit.

The valid values are 0 - 10. The default is 0, which means autorecovery always handles the exit of a node no matter how frequent a node exits.

If **restripeOnDiskFailure** is *no*, **frequentLeaveCountThreshold** has no effect.

frequentLeaveTimespanMinutes
Specifies the time span that is used to calculate the exit frequency of a node. If the exit count of a node within the last **frequentLeaveTimespanMinutes** is greater than the **frequentLeaveCountThreshold**, autorecovery ignores the corresponding node exit.

The valid values are 1 - 1440. The default is 60.

If **restripeOnDiskFailure** is *no*, **frequentLeaveTimespanMinuteshas** no effect.

ignorePrefetchLUNCount
The GPFS client node calculates the number of sequential access prefetch and write-behind threads to run concurrently for each file system by using the count of the number of LUNs in the file system and the value of maxMBpS. However, if the LUNs being used are composed of multiple physical disks, this calculation can underestimate the amount of IO that can be done concurrently.

Setting the value of the ignorePrefetchLUNCount parameter to **yes** does not include the LUN count and uses the maxMBpS value to dynamically determine the number of threads to schedule the prefetchThreads value.

This parameter impacts only the GPFS client node. The GPFS NSD server does not include this parameter in the calculation.

The valid values for this parameter are **yes** and **no**. The default value is **no** and can be used in traditional LUNs where one LUN maps to a single disk or an n+mP array. Set the value of this parameter to **yes** when the LUNs presented to GPFS are made up of a large numbers of physical disks.

The -N flag is valid for this attribute.

ignoreReplicationForQuota
Specifies whether the quota commands ignore data replication factor. Valid values are yes or no. The default value is no.

The **ignoreReplicationForQuota** parameter hides the data replication factor for both input and output quotas commands. This parameter adjusts the values of quota commands according to the data replication factor. For example, if the data replication factor is 2, it means that for

every block in the file, there are effectively 2 blocks being used. For a file of 1 MB, internally 2 MB is allocated. But for the end user, the file size must use 1MB of quota. Without the **ignoreReplicationForQuota** attribute, the quota management reports the file size as 2 MB.

Similarly, quota command inputs are also adjusted with the replication factor. For example, a 1 GB quota limits the overall sum of file sizes to 1 GB, even though internally, the data block usage is 2 GB because of the replication factor.

ignoreReplicationOnStatfs
Specifies whether `df` command output on GPFS file system ignores data replication factor. Valid values are `yes` or `no`. The default value is `no`.

The **ignoreReplicationOnStatfs** parameter ignores the replication factor and helps to report only the actual file size when a **df** command is used. For example, if the data replication factor is 2, it means that for every block in the file, there are effectively 2 blocks being used. For a file of 1 MB, internally 2 MB is allocated. But for the end user, the file size must use 1MB of quota. Without the **ignoreReplicationOnStatfs** attribute, the **df** command reports the file size as 2 MB.

linuxStatfsUnits={posix | subblock | fullblock}
Controls the values that are returned by the Linux functions **statfs** and **statvfs** for **f_bsize**, **f_rsize**, **f_blocks**, and **f_bfree**:

*Table 16. Values returned by statvfs or statfs for different settings of **linuxStatfsUnits***

linuxStatfsUnits	f_bsize	f_frsize	f_blocks	f_bfree
posix	Block size	Subblock size	Units of subblocks	Units of subblocks
subblock	Subblock size	Subblock size	Units of subblocks	Units of subblocks
fullblock	Block size	Block size	Units of blocks	Units of blocks

posix
Returns the correct values as they are specified by POSIX for **statvfs**. This setting might break Linux applications that are written for an earlier version of the **statfs** function and that incorrectly assume that file system capacity (**f_blocks**) and free space (**f_bfree**) are reported in units given by **f_bsize** rather than **f_frsize**.

subblock
Returns values that result in correct disk space requirement calculations but that do not break earlier Linux applications.

fullblock
Returns the same values as do versions of IBM Spectrum Scale that are earlier than 5.0.3. This is the default value.

Note:

- The **posix** value is preferable for applications that use the POSIX-compliant **statvfs** function.
- Linux applications that were built with the earlier Linux **statfs** function might depend on the behavior that is provided by the **fullblock** option.

For more information, see the description of the `-b` *Blocksize* option in the topic "mmcrfs command" on page 318.

The `-N` flag is valid for this attribute.

lrocData
Controls whether user data is populated into the local read-only cache. Other configuration options can be used to select the data that is eligible for the local read-only cache. When using more than one such configuration option, data that matches any of the specified criteria is eligible to be saved.

Valid values are yes or no. The default value is yes.

If lrocData is set to yes, by default the data that was not already in the cache when accessed by a user is subsequently saved to the local read-only cache. The default behavior can be overridden using the lrocDataMaxFileSize and lrocDataStubFileSize configuration options to save all data from small files or all data from the initial portion of large files.

lrocDataMaxFileSize
Limits the data that can be saved in the local read-only cache to only the data from small files.

A value of -1 indicates that all data is eligible to be saved. A value of 0 indicates that small files are not to be saved. A positive value indicates the maximum size of a file to be considered for the local read-only cache. For example, a value of 32768 indicates that files with 32 KB of data or less are eligible to be saved in the local read-only cache. The default value is -1.

lrocDataStubFileSize
Limits the data that can be saved in the local read-only cache to only the data from the first portion of all files.

A value of -1 indicates that all file data is eligible to be saved. A value of 0 indicates that stub data is not eligible to be saved. A positive value indicates that the initial portion of each file that is eligible is to be saved. For example, a value of 32768 indicates that the first 32 KB of data from each file is eligible to be saved in the local read-only cache. The default value is -1.

lrocDirectories
Controls whether directory blocks is populated into the local read-only cache. The option also controls other file system metadata such as indirect blocks, symbolic links, and extended attribute overflow blocks.

Valid values are yes or no. The default value is yes.

lrocEnableStoringClearText
Controls whether encrypted file data can be read into a local read-only cache (LROC) device. Valid values are yes and no. The default value is no.

If the value is yes, encrypted files can benefit from the performance improvements that are provided by an LROC device. However, be aware that IBM Spectrum Scale holds encrypted file data in memory as cleartext. Because LROC storage is non-volatile, an attacker can capture the cleartext by removing the LROC device from the system and reading the contents at some other location.

Warning: You must take steps to protect the cleartext while it is in LROC device storage. One method is to install an LROC device that internally encrypts data that is written into it and decrypts data that is read from it. However, be aware that a device of this type voids the IBM Spectrum Scale secure deletion guarantee, because IBM Spectrum Scale does not manage the encryption key for the device.

For more information, see the following topics:

Encryption and local read-only cache (LROC) in the *IBM Spectrum Scale: Administration Guide*.
Local read-only cache in the *IBM Spectrum Scale: Administration Guide*.

lrocInodes
Controls whether inodes from open files is populated into the local read-only cache; the cache contains the full inode, including all disk pointers, extended attributes, and data.

Valid values are yes or no. The default value is yes.

logRecoveryThreadsPerLog
Controls the number of threads that are available for recovering a single log file. The default value is 8 and the valid range is 1 - 64. Setting a higher value expedites the recovery of single log files that are being replayed. The improvement in processing speed depends on the log file size and user workload.

logOpenParallelism
Controls the number of log files that can be opened in parallel during a log recovery. The default value is 8 and the valid range is 1 - 256. Setting a higher value improves the recovery speed of the file system when it is mounted on multiple nodes.

logRecoveryParallelism
Controls the number of log files that can be recovered concurrently. The default value is 1 and the valid range is 1 - 64. Setting a higher value expedites the recovery of the file system when multiple nodes fail at the same time.

maxActiveIallocSegs
Specifies the number of active inode allocation segments that are maintained on the specified nodes. The valid range is 1 - 64. A value greater than 1 can significantly improve performance in the following scenario:

1. A single node has created a large number of files in multiple directories.
2. Processes and threads on multiple nodes are now concurrently attempting to delete or unlink files in those directories.

Values greater than 8 might not provide any improvement in performance.

A value greater than 1 is supported only on file systems that are created at or upgraded to file system format version 5.0.2 or later (file system format number 20.00 or later).

The default value is 8 on file systems that are created at file system format version 5.0.2 or later. The default value is 1 on earlier file system format versions and on file systems that are upgraded to file system format version 5.0.2.

A change in the value of this attribute is not effective until after the file system is remounted.

If the value of this attribute has not been changed since the file system was created, the `mmlsconfig` command lists the value as -1. But the actual value is 1 or 8, depending on the level of the file system:

- File systems created at version 5.0.2 or later: 8
- File systems created at an earlier version: 1

The -N flag is valid for this attribute.

maxblocksize
Changes the maximum file system block size. Valid block sizes are 64 KiB, 128 KiB, 256 KiB, 512 KiB, 1 MiB, 2 MiB, 4 MiB, 8 MiB, and 16 MiB. Specify this value with the character K or M; for example, use 8M to specify a block size of 8 MiB. When you create a new cluster, `maxblocksize` is set to DEFAULT (4 MiB). For more information, see “mmcrfs command” on page 318.

File systems with block sizes larger than the specified value cannot be created or mounted unless the block size is increased.

GPFS must be down on all the nodes in the cluster when you change the `maxblocksize` attribute.

Note: When you migrate a cluster from an earlier version to 5.0.0 or later, the value of `maxblocksize` stays the same. However, if `maxblocksize` was set to DEFAULT in the earlier version of the cluster, then migrating it to 5.0.0 or later sets it explicitly to 1 MiB, which was the default value in earlier versions. To change maxblocksize to the default value after migrating to 5.0.0 or later, set `maxblocksize=DEFAULT` (4 MiB).

maxBufferDescs
Valid values are from 512 to 10,000,000.

Without explicit setting, it is set to a value of 10 * `maxFilesToCache` up to `pagepool` size/16 KB. Each buffer descriptor caches maximum block size data for a file. When caching small files, it does not need to be more than a small multiple of `maxFilesToCache` since only OpenFile objects can cache data blocks. When an application needs to cache large files, `maxBufferDescs` can be tuned to ensure that there are enough to cache large files.

For example, if you have 10,000 buffer descriptors that are configured and a 1MiB file system block size, you do not have enough buffer descriptors to cache a 20 GiB file. To cache a 20 GiB file, increase `maxBufferDescs` to at least 20,480 (20 GiB/1MiB=20,480).

The `-N` flag is valid for this attribute.

Note: When LROC is configured on the node, additional buffer descriptors may be required to reference data that is stored in LROC. For more information, see *Local read-only cache* in *IBM Spectrum Scale: Administration Guide*.

maxDownDisksForRecovery
Specifies the maximum number of disks that might experience a failure and still be subject to an automatic recovery attempt. If this value is exceeded, no automatic recovery actions take place.

Valid values are in the range 0 - 300. The default is 16. If `restripeOnDiskFailure` is no, `maxDownDisksForRecovery` has no effect.

maxFailedNodesForRecovery
Specifies the maximum number of nodes that might be unavailable before automatic disk recovery actions are canceled.

Valid values are in the range 0 - 300. The default is 3. If `restripeOnDiskFailure` is no, `maxFailedNodesForRecovery` has no effect.

maxFcntlRangesPerFile
Specifies the number of **fcntl** locks that are allowed per file. The default is 200. The minimum value is 10 and the maximum value is 200000.

maxFilesToCache
Specifies the number of inodes to cache for open files or files that are recently closed. This parameter does not limit the number of files that can remain concurrently open on the node.

Storing the inode of a file in cache permits faster re-access to the file. The default is 4000, but increasing this number might improve throughput for workloads with high file reuse. However, increasing this number excessively might cause paging at the file system manager node. The value should be large enough to handle the number of concurrently open files plus allow caching of recently used files.

The `-N` flag is valid for this attribute.

maxMBpS
Specifies an estimate of how many megabytes of data can be transferred per second into or out of a single node. The default is 2048 MiB per second. The value is used in calculating the amount of I/O that can be done to effectively prefetch data for readers and write-behind data from writers. By lowering this value, you can artificially limit how much I/O one node can put on all of the disk servers.

The `-N` flag is valid for this attribute.

maxMissedPingTimeout
See the `minMissedPingTimeout` parameter.

maxReceiverThreads
Controls the maximum number of receiver threads that handle incoming TCP packets. The actual number of receiver threads that are configured is limited by the number of logical CPUs on the node. If the number of logical CPUs is less than **maxReceiverThreads**, then the number of threads that are handling the packets is set to the number of logical CPUs.

The default value of **maxReceiverThreads** is 32. The maximum value of **maxReceiverThreads** is 128. Range is 1-128.

This parameter should be increased on large clusters to limit the number of sockets that are handled by each receive thread. For clusters with 2048 nodes or more, set the parameter to the number of logical CPUs.

The -N flag is valid for this attribute.

maxStatCache
Specifies the number of inodes to keep in the stat cache. The stat cache maintains only enough inode information to perform a query on the files kept in the stat cache as would be needed by the 'ls' command. The valid range for maxStatCache is 0 - 100,000,000.

The default value of the maxStatCache parameter depends on the value of maxFilesToCache parameter in certain scenarios. The following table provides examples of such scenarios:

Table 17. Default value of maxStatCache parameter

Scenarios	maxFilesToCache	maxStatCache	Comments
No changes to the default values of maxFilesToCache and maxStatCache	4000	1000	These default values are applied.
Value of maxFilesToCache is less than 2500	[< 2500]	4 * maxFilesToCache	Example: If maxFilesToCache is 2000, then maxStatCache is 8000.
Value of maxFilesToCache is 2500 or higher	[>= 2500]	10000	If maxStatCache is calculated by the daemon, the value is capped at 10000.

If you do not accept either of the default values, set maxStatCache to an appropriate size based on the number of nodes in the cluster, the number of token managers in the cluster, the size of the Local Read-Only Cache (LROC) if one is configured, and any other relevant factors.

Note: In versions of IBM Spectrum Scale earlier than 5.0.2, the stat cache is not effective on the Linux platform unless the LROC is configured. In versions earlier than 5.0.2, follow these guidelines:

- If LROC is not enabled on the node, set maxStatCache to 0.
- If LROC is enabled on the node, accept a default value of maxStatCache or set maxStatCache to an appropriate size as described in the previous paragraphs.

This pre-5.0.2 restriction applies to all the versions and distributions of Linux that IBM Spectrum Scale supports.

The -N flag is valid for this attribute.

maxTcpConnsPerNodeConn
Controls the maximum number of TCP connections that the GPFS mmfsd daemon will establish to another node. Valid values are 1-8, with the default being 2. For any given pair of nodes, the number of established TCP connections will be the minimum value of **maxTcpConnsPerNodeConn** defined on these two nodes. For further information on configuring this parameter, see *Recommendations for tuning maxTcpConnsPerNodeConn parameter* in *IBM Spectrum Scale: Administration Guide*.

The -N flag is valid for this attribute.

Note: In IBM Spectrum Scale 5.1.1 or later, multiple TCP connections may be used to communicate with another remote GPFS daemon. Although a single destination IP address is always used, multiple physical links may be used for those connections given the appropriate configuration. For example, an 802.3ad connection with a layer 3+4 xmit_hash_policy.

metadataDiskWaitTimeForRecovery
Specifies a period, in seconds, during which the recovery of metadata disks is suspended to give the disk subsystem a chance to correct itself. This parameter is taken into account when the affected disks belong to a single failure group. If more than one failure group is affected, the delay is based on the value of `minDiskWaitTimeForRecovery`.

Valid values are 0 - 3600 seconds. The default is 2400. If `restripeOnDiskFailure` is no, `metadataDiskWaitTimeForRecovery` has no effect.

minDiskWaitTimeForRecovery
Specifies a period, in seconds, during which the recovery of disks is suspended to give the disk subsystem a chance to correct itself. This parameter is taken into account when more than one failure group is affected. If the affected disks belong to a single failure group, the delay is based on the values of `dataDiskWaitTimeForRecovery` and `metadataDiskWaitTimeForRecovery`.

Valid values are 0 - 3600 seconds. The default is 1800. If `restripeOnDiskFailure` is no, `minDiskWaitTimeForRecovery` has no effect.

minIndBlkDescs
Specifies the total number of indirect blocks in cache where the disk addresses of data blocks or indirect blocks of files are stored. Each indirect block descriptor caches one indirect block. Caching these indirect blocks enable faster retrieval of the disk location of the data to be read or written, thus improving the performance of I/Os.

The default value for `minIndBlkDesc` is 5000. The `minIndBlkDesc` value is calculated by assigning the maximum value between `minIndBlkDesc` and `maxFilesToCache`.

For example:

- If user does not set the value for `minIndBlkDesc`, then its effective value is MAX (5000,`maxFilesToCache`), whichever is the higher.
- If user sets the value for `minIndBlkDesc`, then its effective value is MAX (`minIndBlkDesc`, `maxFilesToCache`), whichever is the higher.

The `minIndBlkDesc` value must be large enough to handle the number of concurrently open files, or traverse many data blocks in large files, if the `maxFilesToCache` is set to a small value.

The `-N` flag is valid for this attribute.

minMissedPingTimeout
The `minMissedPingTimeout` and `maxMissedPingTimeout` parameters set limits on the calculation of `missedPingTimeout` (MPT). The MPT is the allowable time for pings sent from the Cluster Manager (CM) to a node that has not renewed its lease to fail. The default MPT value is 5 seconds less than `leaseRecoveryWait`. The CM will wait the MPT seconds after the lease has expired before declaring a node out of the cluster. The values of the `minMissedPingTimeout` and `maxMissedPingTimeout` are in seconds; the default values are 3 and 60 respectively. If these values are changed, only GPFS on the quorum nodes that elect the CM must be recycled to take effect.

This parameter can be used to cover over a central network switch failure timeout or other network glitches that might be longer than `leaseRecoveryWait`. This might prevent false node down conditions, but it extends the time for node recovery to finish and might block other nodes from progressing if the failing node holds the tokens for many shared files.

As is the case with `leaseRecoveryWait`, a node is usually expelled from the cluster if there is a problem with the network or the node runs out of resources like paging. For example, if there is an application that is running on a node that is paging the machine too much or overrunning network capacity, GPFS might not have the chance to contact the Cluster Manager node to renew the lease within the timeout period.

The default value of this parameter is 3. A valid value is any number in the range 1 - 300.

mmapRangeLock
Specifies POSIX or non-POSIX `mmap` byte-range semantics. Valid values are `yes` or `no` (`yes` is the default). A value of `yes` indicates POSIX byte-range semantics apply to `mmap` operations. A value of `no` indicates non-POSIX `mmap` byte-range semantics apply to `mmap` operations.

If using InterProcedural Analysis (IPA), turn off this option:

```
mmchconfig mmapRangeLock=no -i
```

This allows more lenient intranode locking, but imposes internode whole file range tokens on files using `mmap` while writing.

mmfsLogTimeStampISO8601
Controls the time stamp format for GPFS log entries. Specify `yes` to use the ISO 8601 time stamp format for log entries or `no` to use the earlier time stamp format. The default value is `yes`. You can specify the log time stamp format for the entire cluster or for individual nodes. You can have different log time stamp formats on different nodes of the cluster. For more information, see the *Time stamp in GPFS log entries* topic in *IBM Spectrum Scale: Problem Determination Guide*.

The `-N` flag is valid for this attribute. This attribute takes effect immediately, whether or not `-i` is specified.

nfsPrefetchStrategy
With the `nfsPrefetchStrategy` parameter, GPFS optimizes prefetching for NFS file-style access patterns. This parameter defines a window of the number of blocks around the current position that are treated as fuzzy-sequential access. The value of this parameter can improve the performance while reading large files sequentially. However, because of kernel scheduling, some read requests that come to GPFS are not sequential. If the file system block size is smaller than the read request sizes, increasing the value of this parameter provides a bigger window of blocks. The default value is 0. A valid value is any number in the range 0 - 10.

Setting the value of `nfsPrefetchStrategy` to 1 or greater can improve the sequential read performance when large files are accessed by using NFS and the file system block size is smaller than the NFS transfer block size.

nistCompliance
Controls whether GPFS operates in the NIST 800-131A mode. (This applies to security transport only, not to encryption, as encryption always uses NIST-compliant mechanisms.)

Valid values are:

off
Specifies that there is no compliance to NIST standards. For clusters that are operating below the GPFS 4.1 level, this is the default. For clusters at the version 5.1 level or higher, setting **`nistCompliance`** to `off` is not allowed.

SP800-131A
Specifies that security transport is to follow the NIST SP800-131A recommendations. For clusters at the GPFS 4.1 level or higher, this is the default.

Note: In a remote cluster setup, all clusters must have the same `nistCompliance` value.

noSpaceEventInterval
Specifies the time interval between calling a callback script of two `noDiskSpace` events of a file system. The default value is 120 seconds. If this value is set to zero, the `noDiskSpace` event is generated every time the file system encounters the `noDiskSpace` event. The `noDiskSpace` event is generated when a callback script is registered for this event with the `mmaddcallback` command.

nsdBufSpace
This option specifies the percentage of the page pool that is reserved for the network transfer of NSD requests. Valid values are within the range of 10 to 70. The default value is 30. On IBM Spectrum Scale RAID recovery group NSD servers, this value should be decreased to its minimum of 10, since vdisk-based NSDs are served directly from the RAID buffer pool (as governed by

nsdRAIDBufferPoolSizePct). On all other NSD servers, increasing either this value or the amount of page pool, or both, could improve NSD server performance. On NSD client-only nodes, this parameter is ignored. For more information about IBM Spectrum Scale RAID, see *IBM Spectrum Scale RAID: Administration*.

The -N flag is valid for this attribute.

nsdCksumTraditional
This attribute enables checksum data-integrity checking between a traditional NSD client node and its NSD server. Valid values are yes and no. The default value is no. (*Traditional* in this context means that the NSD client and server are configured with IBM Spectrum Scale rather than with IBM Spectrum Scale RAID. The latter is a component of IBM Elastic Storage Server (ESS) and of IBM GPFS Storage Server (GSS).)

The checksum procedure detects any corruption by the network of the data in the NSD RPCs that are exchanged between the NSD client and the server. A checksum error triggers a request to retransmit the message.

When this attribute is enabled on a client node, the client indicates in each of its requests to the server that it is using checksums. The server uses checksums only in response to client requests in which the indicator is set. A client node that accesses a file system that belongs to another cluster can use checksums in the same way.

You can change the value of the this attribute for an entire cluster without shutting down the mmfsd daemon, or for one or more nodes without restarting the nodes.

Note:

- Enabling this feature can result in significant I/O performance degradation and a considerable increase in CPU usage.
- To enable checksums for a subset of the nodes in a cluster, issue a command like the following one:

```
mmchconfig nsdCksumTraditional=yes -i -N <subset-of-nodes>
```

The -N flag is valid for this attribute.

nsdDumpBuffersOnCksumError
This attribute enables the dumping of the data buffer to a file when a checksum error occurs. Valid values are yes and no. The default value is no. The location of the dump file is set by the dataStructureDump attribute.

You can change the value of the nsdDumpBuffersOnCksumError attribute for a cluster without shutting down the mmfsd daemon, or for one or more nodes without restarting the nodes.

The -N flag is valid for this attribute.

nsdInlineWriteMax
The nsdInlineWriteMax parameter specifies the maximum transaction size that can be sent as embedded data in an NSD-write RPC. The value of this parameter is ignored if verbs send is enabled by using the verbsRdmaSend parameter.

In most cases, the NSD-write RPC exchange performs the following steps:

1. An RPC is sent from the client to the server to request a write.
2. A GetData RPC is sent back from the server to the client to request the data.

Note: For data smaller than nsdInlineWriteMax, GPFS sends that amount of write data directly without the GetData RPC from the server to the client.

The default value of this parameter is 1024. A valid value is any number in the range 0 - 8M.

nsdMaxWorkerThreads
The nsdMaxWorkerThreads parameter sets the maximum number of NSD threads that can be involved in NSD I/O operations on an NSD server to the storage system to which the

server is connected. On 64-bit architectures, the maximum value of the workerThreads, prefetchThreads, and nsdMaxWorkerThreads is less than 8192. The minimum value of nsdMaxWorkerThreads is 8 and the default value is 512. The default value works for most of the use cases.

This value should be scaled with the number of NSDs the server is connected with and not NSD clients in the cluster. When the NSDs to which the server is doing I/O are constructed from high performance storage device, such as IBM FlashSystem® or ESS, increased value of nsdMaxWorkerThreads can obtain better performance.

nsdMinWorkerThreads
The nsdMinWorkerThreads parameter sets a lower bound on number of active NSD I/O threads on an NSD server node that executes I/O operations against NSDs. The value represents the minimum number of NSD I/O threads required to execute the I/O operations.

The default value of this parameter is 16 and the minimum value is 1. For limits on setting the number of NSD threads, see the description of nsdMaxWorkerThreads.

nsdMultiQueue
The nsdMultiQueue parameter sets the number of queues. The default value of this parameter is 256. A valid value is any number in the range 2 - 512.

nsdRAIDBufferPoolSizePct
This option specifies the percentage of the page pool that is used for the IBM Spectrum Scale RAID vdisk buffer pool. Valid values are within the range of 10 to 90. The default is 50 when IBM Spectrum Scale RAID is configured on the node in question; 0 when it is not. For more information about IBM Spectrum Scale RAID, see *IBM Spectrum Scale RAID: Administration*.

The -N flag is valid for this attribute.

nsdRAIDTracks
This option specifies the number of tracks in the IBM Spectrum Scale RAID buffer pool, or 0 if this node does not have a IBM Spectrum Scale RAID vdisk buffer pool. This controls whether IBM Spectrum Scale RAID services are configured. For more information about IBM Spectrum Scale RAID, see *IBM Spectrum Scale RAID: Administration*.

Valid values are: 0; 256 or greater.

The -N flag is valid for this attribute.

nsdServerWaitTimeForMount
When mounting a file system whose disks depend on NSD servers, this option specifies the number of seconds to wait for those servers to come up. The decision to wait is controlled by the criteria managed by the nsdServerWaitTimeWindowOnMount option.

Valid values are 0 - 1200 seconds. The default is 300. A value of zero indicates that no waiting is done. The interval for checking is 10 seconds. If nsdServerWaitTimeForMount is 0, nsdServerWaitTimeWindowOnMount has no effect.

The mount thread waits when the daemon delays for safe recovery. The mount wait for NSD servers to come up, which is covered by this option, occurs after expiration of the recovery wait allows the mount thread to proceed.

The -N flag is valid for this attribute.

nsdServerWaitTimeWindowOnMount
Specifies a window of time (in seconds) during which a mount can wait for NSD servers as described for the nsdServerWaitTimeForMount option. The window begins when quorum is established (at cluster startup or subsequently), or at the last known failure times of the NSD servers required to perform the mount.

Valid values are 1 - 1200 seconds. The default is 600. If nsdServerWaitTimeForMount is 0, nsdServerWaitTimeWindowOnMount has no effect.

The -N flag is valid for this attribute.

When a node rejoins the cluster after being removed for any reason, the node resets all the failure time values that it knows about. Therefore, when a node rejoins the cluster it believes that the NSD servers have not failed. From the perspective of a node, old failures are no longer relevant.

GPFS checks the cluster formation criteria first. If that check falls outside the window, GPFS then checks for NSD server fail times being within the window.

numaMemoryInterleave
In a Linux NUMA environment, the default memory policy is to allocate memory from the local NUMA node of the CPU from which the allocation request was made. This parameter is used to change to an interleave memory policy for GPFS by starting GPFS with `numactl --interleave=all`.

Valid values are `yes` and `no`. The default is `no`.

If you run IBM Spectrum Scale on a multi-processor system, you should set this variable to `yes` and restart GPFS.

Before using this parameter, ensure that the Linux `numactl` package has been installed.

pagepool
Changes the size of the cache on each node. The default value is either one-third of the physical memory on the node or 1 GiB, whichever is smaller. This applies to new installations only; on upgrades the existing default value is kept.

The maximum GPFS page pool size depends on the value of the `pagepoolMaxPhysMemPct` parameter and the amount of physical memory on the node. You can specify this value with the suffix `K`, `M`, or `G`, for example, `128M`.

If a node in the cluster is an NSD server or is configured for IBM Spectrum Scale RAID, this parameter takes effect on the node only when the GPFS daemon is restarted, even if the `-i` or `-I` option is specified.

The `-N` flag is valid for this attribute.

pagepoolMaxPhysMemPct
Percentage of physical memory that can be assigned to the page pool. Valid values are 10 - 90 percent. The default is 75 percent (with the exception of Windows, where the default is 50 percent).

The `-N` flag is valid for this attribute.

panicOnIOHang={yes | no}
Controls whether the GPFS daemon panics the node kernel when a local I/O request is pending in the kernel for more than five minutes. This attribute applies only to disks that the node is directly attached to.

yes
Causes the GPFS daemon to panic the node kernel.

no
Takes no action. This is the default value.

This attribute is not supported in the Microsoft Windows environment.

Note: With the **diskIOHang** event of the **mmaddcallback** command, you can add notification and data collection scripts to isolate the reason for a long I/O wait.

The `-N` flag is valid for this attribute.

pitWorkerThreadsPerNode
Controls the maximum number of threads to be involved in parallel processing on each node that is serving as a Parallel Inode Traversal (PIT) worker.

By default, when a command that uses the PIT engine is run, the file system manager asks all nodes in the local cluster to serve as PIT workers; however, you can specify an exact set of nodes to serve as PIT workers by using the `-N` option of a PIT command. Note that the

current file system manager node is a mandatory participant, even if it is not in the list of nodes you specify. On each participating node, up to `pitWorkerThreadsPerNode` can be involved in parallel processing. The range of accepted values is 0 to 8192. The default value varies within the 2-16 range, depending on the file system configuration. If a file system contains vdisk-based NSD disks, the default value varies within the 8-64 range.

prefetchPct
GPFS uses the `prefetchPct` parameter as a guideline to limit the page pool space that is to be used for prefetch and write-behind buffers for active sequential streams. The default value of the `prefetchPct` parameter is 20% of the `pagepool` value. If the workload is sequential with very little caching of small files or random IO, increase the value of this parameter to 60% of the `pagepool` value, so that each stream can have more buffers that are cached for prefetch and write-behind operations.

The default value of this parameter is 20. The valid value can be any numbers in the range 0 - 60.

prefetchThreads
Controls the maximum number of threads that are dedicated to prefetching data for files that are read sequentially, or to handle sequential write-behind.

Functions in the GPFS daemon dynamically determine the actual degree of parallelism for prefetching data. The default value is 72. The minimum value is 2. The maximum value of `prefetchThreads` plus `worker1Threads` plus `nsdMaxWorkerThreads` is 8192 on all 64-bit platforms.

The `-N` flag is valid for this attribute.

proactiveReconnect={yes | no}
When enabled causes nodes to proactively close problematic TCP connections with other nodes and to reestablish new connections in their place. The default value of the **proactiveReconnect** parameter is 'no' when the minimum release level of a cluster is less than 5.0.4. The default value of the **proactiveReconnect** parameter is 'yes' when the minimum release level of a cluster is at least 5.0.4.

yes

Each node monitors the state of TCP connections with other nodes in the cluster and proactively closes and reestablishes a connection whenever the TCP congestion state and TCP retransmission timeout (RTO) (similar to the information that is displayed by the **ss -i** command in Linux) indicate that data might not be flowing.

In certain environments that are prone to short-term network outages, this feature can prevent nodes from being expelled from a cluster when TCP connections go into error states that are caused by packet loss in the network or in the adapter. If a TCP connection is successfully reestablished and operates normally, nodes on either side of the connection are not expelled.

no
Disables proactive closing and reestablishing of problematic TCP connections between nodes.

For both the `yes` and `no` settings, a message is written to the `mmfs.log` file whenever a TCP connection reaches an error state.

This attribute is supported only on Linux.

The `-N` flag is valid for this attribute.

profile
Specifies a predefined profile of attributes to be applied. System-defined profiles are located in /usr/lpp/mmfs/profiles/. All the configuration attributes listed under a cluster stanza are changed as a result of this command. The following system-defined profile names are accepted:

- `gpfsProtocolDefaults`
- `gpfsProtocolRandomIO`

A user's profiles must be installed in /var/mmfs/etc/. The profile file specifies GPFS configuration parameters with values different than the documented defaults. A user-defined profile must not begin with the string 'gpfs' and must have the .profile suffix.

User-defined profiles consist of the following stanzas:

```
%cluster:
[CommaSeparatedNodesOrNodeClasses:]ClusterConfigurationAttribute=Value
...
```

File system attributes and values are ignored.

A sample file can be found in /usr/lpp/mmfs/samples/sample.profile. See the **mmchconfig** command for a detailed description of the different configuration parameters. User-defined profiles should be used only by experienced administrators. When in doubt, use the **mmchconfig** command instead.

readReplicaPolicy
Specifies the location from which the disk is to read replicas. By default, GPFS reads the first replica whether there is a replica on the local disk or not. When `readReplicaPolicy=local` is specified, the policy reads replicas from the local disk if the local disk has data; for performance considerations, this is the recommended setting for FPO environments. An NSD server on the same subnet as the client is also considered to be local. When the subnets configuration parameter is set, all configured subnets are taken into account; not merely the daemon IP address.

When `readReplicaPolicy=fastest` is specified, the policy reads replicas from the disk considered the fastest based on the read I/O statistics of the disk. You can tune the way the system determines the fastest policy using the following parameters:

- `fastestPolicyNumReadSamples`
- `fastestPolicyCmpThreshold`
- `fastestPolicyMaxValidPeriod`
- `fastestPolicyMinDiffPercent`

In a system with SSD and regular disks, the value of `fastestPolicyCmpThreshold` can be set to a greater number to let GPFS refresh the speed statistics for the slower disks less frequently. The default value is maintained for all other configurations. The default value of this parameter is default. The valid values are default, local, and fastest.

To return this attribute to the default setting, specify `readReplicaPolicy=DEFAULT -i`.

readReplicaRuleEnabled
Specifies if the `gpfs.readReplicaRule` extended attribute or the `diskReadExclusionList` configuration option are evaluated during the data block read. The valid values are yes and no. The default value is no. For more information, see the *Replica mismatches* topic in the *IBM Spectrum Scale: Problem Determination Guide*.

release=LATEST

Increases the minimum release level of a cluster to the latest version of IBM Spectrum Scale that is supported by all the nodes of the cluster. For example, if the minimum release level of a cluster is 5.0.1.3 but IBM Spectrum Scale 5.0.2.2 is installed on all the nodes, then **release=LATEST** increases the minimum release level of the cluster to 5.0.2.0.

The effect of increasing the minimum release level is to enable the features that are installed with the version of IBM Spectrum Scale that the new minimum release level specifies. To return to the preceding example, increasing the minimum release level to 5.0.2.0 enables the features that are installed with IBM Spectrum Scale 5.0.2.0.

Issuing `mmchconfig` with the **release=LATEST** parameter is one of the final steps in upgrading the nodes of a cluster to a later version of IBM Spectrum Scale. For more information, see the

topic *Completing the upgrade to a new level of IBM Spectrum Scale* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Before you use this parameter, consider any possible unintended consequences. For more information read the help topic *Minimum release level of a cluster* in the *IBM Spectrum Scale: Administration Guide*.

To process this parameter, the `mmchconfig` command must access each node in the cluster to determine the version of IBM Spectrum Scale that is installed. If the command cannot access one or more of the nodes, it displays an error message and terminates. You must correct the communication problem and issue the command again. Repeat this process until the command verifies the information for all the nodes and ends successfully.

This parameter causes the `mmchconfig` command to fail with an error message if the `cipherList` configuration attribute of the cluster is not set to `AUTHONLY` or higher. For more information, see the topic *Completing the migration to a new level of IBM Spectrum Scale* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

To display the minimum release level, issue the following command:

```
mmlsconfig minReleaseLevel
```

restripeOnDiskFailure
Specifies whether GPFS will attempt to automatically recover from certain common disk failure situations.

When a disk experiences a failure and becomes unavailable, the recovery procedure first attempts to restart the disk and if this fails, the disk is suspended and its data that is moved to other disks. Similarly, when a node joins the cluster, all disks for which the node is responsible are checked and an attempt is made to restart any that are in a down state.

Whether a file system is a subject of a recovery attempt is determined by the max replication values for the file system. If the `mmlsfs -M` or `-R` value is greater than one, then the recovery code is executed. The recovery actions are asynchronous and GPFS continues its processing while the recovery attempts take place. The results from the recovery actions and any errors that are encountered is recorded in the `/var/adm/ras/autorecovery.log.<timestamp>` log.

For more information about GPFS disk fail auto recovery, see *Auto recovery* in *IBM Spectrum Scale: Big Data and Analytics Guide*.

rpcPerfNumberDayIntervals
Controls the number of days that aggregated RPC data is saved. Every day the previous 24 hours of one-hour RPC data is aggregated into a one-day interval.

The default value for `rpcPerfNumberDayIntervals` is 30, which allows the previous 30 days of one-day intervals to be displayed. To conserve memory, fewer intervals can be configured to reduce the number of recent one-day intervals that can be displayed. The values that are allowed for `rpcPerfNumberDayIntervals` are in the range 4 - 60.

rpcPerfNumberHourIntervals
Controls the number of hours that aggregated RPC data is saved. Every hour the previous 60 minutes of 1-minute RPC data is aggregated into a one-hour interval.

The default value for `rpcPerfNumberHourIntervals` is 24, which allows the previous day's worth of one-hour intervals to be displayed. To conserve memory, fewer intervals can be configured to reduce the number of recent one-hour intervals that can be displayed. The values that are allowed for `rpcPerfNumberHourIntervals` are 4, 6, 8, 12, or 24.

rpcPerfNumberMinuteIntervals
Controls the number of minutes that aggregated RPC data is saved. Every minute the previous 60 seconds of 1-second RPC data is aggregated into a 1-minute interval.

The default value for `rpcPerfNumberMinuteIntervals` is 60, which allows the previous hour's worth of 1-minute intervals to be displayed. To conserve memory, fewer intervals can be

configured to reduce the number of recent 1-minute intervals that can be displayed. The values that are allowed for `rpcPerfNumberMinuteIntervals` are 4, 5, 6, 10, 12, 15, 20, 30, or 60.

rpcPerfNumberSecondIntervals
Controls the number of seconds that aggregated RPC data is saved. Every second RPC data is aggregated into a 1-second interval.

The default value for `rpcPerfNumberSecondIntervals` is 60, which allows the previous minute's worth of 1-second intervals to be displayed. To conserve memory, fewer intervals can be configured to reduce the number of recent 1-second intervals that can be displayed. The values that are allowed for `rpcPerfNumberSecondIntervals` are 4, 5, 6, 10, 12, 15, 20, 30, or 60.

rpcPerfRawExecBufferSize
Specifies the number of bytes to allocate for the buffer that is used to store raw RPC execution statistics. For each RPC received by a node, 16 bytes of associated data is saved in this buffer when the RPC completes. This circular buffer must be large enough to hold 1 second's worth of raw execution statistics.

The default value for `rpcPerfRawExecBufferSize` is 10 MiB, which produces 655360 entries. The data in this buffer is processed every second. It is a good idea to set the buffer size 10% to 20% larger than what is needed to hold 1 second's worth of data.

rpcPerfRawStatBufferSize
Specifies the number of bytes to allocate for the buffer that is used to store raw RPC performance statistics. For each RPC sent to another node, 56 bytes of associated data is saved in this buffer when the reply is received. This circular buffer must be large enough to hold 1 second's worth of raw performance statistics.

The default value for `rpcPerfRawStatBufferSize` is 30 MiB, which produces 561737 entries. The data in this buffer is processed every second. It is a good idea to set the buffer size 10% to 20% larger than what is needed to hold one second's worth of data.

sdrNotifyAuthEnabled
Specifies whether to authenticate the notify RPCs that are related to deadlock detection and amelioration, node overload, and node expel. Possible values are `yes` or `no`. The value `yes` forces to authenticate such notify RPCs.

For new clusters with a minimum release level of 5.1.1 or later, the default value of this parameter is `yes`. For existing clusters, the default value of this parameter is `no` and the system administrator can change it to `yes`, if the cluster uses CCR and the minimum release level is at least 5.1.1.

The scope of the **sdrNotifyAuthEnabled** parameter is local to a cluster. However, when set to `yes`, it can impact the notify RPCs that are sent to remote clusters. The behavior of the notify RPCs varies based on the value of **sdrNotifyAuthEnabled** parameter in different configurations. For example:

1. If **sdrNotifyAuthEnabled** is set to different values in remote clusters, then the notify RPCs sent between the remote clusters fails and an error message is logged in the `mmfs.log` file.
2. The sender of the notify RPC has **sdrNotifyAuthEnabled** set to `yes` and the receiver of the notify RPC has **sdrNotifyAuthEnabled** set to `no`. In this case, the sender of the notify RPC returns an error and logs an error in the message log, as it cannot establish a secure connection, but the receiver will service the notify RPC request.
3. The sender of the notify RPC has **sdrNotifyAuthEnabled** set to `no` and the receiver of the notify RPC has **sdrNotifyAuthEnabled** set to `yes`. In this case, the sender of the notify RPC does not return an error and the receiver of the notify RPC does not service the RPC and logs an error message in the message log.

seqDiscardThreshold
With the `seqDiscardThreshold` parameter, GPFS detects a sequential read or write access pattern and specifies what has to be done with the page pool buffer after it is consumed or flushed by write-behind threads. This is the highest performing option in a case where a very large file is read or written sequentially. The default for this value is 1 MiB, which means that

if a file is sequentially read and is greater than 1 MiB, GPFS does not keep the data in cache after consumption. There are some instances where large files are reread by multiple processes such as data analytics. In some cases, you can improve the performance of these applications by increasing the value of the `seqDiscardThreshold` parameter so that it is larger than the sets of files that have to be cached. If the value of the `seqDiscardthreshold` parameter is increased, GPFS attempts to keep as much data in cache as possible for the files that are below the threshold.

The value of `seqDiscardThreshold` is file size in bytes. The default is 1/ MB. Increase this value if you want to cache files that are sequentially read or written and are larger than 1 MiB in size. Ensure that there are enough buffer descriptors to cache the file data. For more information about buffer descriptors, see the `maxBufferDescs` parameter.

sharedTmpDir
Specifies a default global work directory where the `mmapplypolicy` command or the `mmbackup` command can store the temporary files that it generates during its processing. The command uses this directory when no global work directory was specified on the command line with the `-g` option. The directory must be in a file system that meets the requirements of the `-g` option. For more information, see “mmapplypolicy command” on page 80.

Note: The `mmapplypolicy` command or the `mmbackup` command uses this directory regardless of the format version of the target file system. That is, to take advantage of this attribute, you do not need to upgrade your file system to file system format version 5.0.1 or later (file system format number 19.01 or greater).

sidAutoMapRangeLength
Controls the length of the reserved range for Windows SID to UNIX ID mapping. See *Identity management on Windows* in the *IBM Spectrum Scale: Administration Guide* for additional information.

sidAutoMapRangeStart
Specifies the start of the reserved range for Windows SID to UNIX ID mapping. See *Identity management on Windows* in the *IBM Spectrum Scale: Administration Guide* for additional information.

subnets
Specifies subnets that are used to communicate between nodes in a GPFS cluster or a remote GPFS cluster.

The subnets option must use the following format:

```
subnets="Subnet[/ClusterName[;ClusterName...][ Subnet[/ClusterName[;ClusterName...]...]"
```

where:

Subnet
Is a subnet specification such as `192.168.2.0`.

ClusterName
Can be either a cluster name or a shell-style regular expression, which is used to match cluster names, such as:

CL[23].kgn.ibm.com
Matches `CL2.kgn.ibm.com` and `CL3.kgn.ibm.com`.

CL[0-7].kgn.ibm.com
Matches `CL0.kgn.ibm.com`, `CL1.kgn.ibm.com`, ... `CL7.kgn.ibm.com`.

CL*.ibm.com
Matches any cluster name that starts with `CL` and ends with `.ibm.com`.

CL?.kgn.ibm.com
Matches any cluster name that starts with `CL`, is followed by any one character, and then ends with `.kgn.ibm.com`.

The order in which you specify the subnets determines the order in which GPFS uses these subnets to establish connections to the nodes within the cluster. GPFS follows the network settings of the operating system for a specified subnet address, including the network mask. For example, if you specify `subnets="192.168.2.0"` and a 23-bit mask is configured, then the subnet spans IP addresses `192.168.2.0 - 192.168.3.255`. In contrast, with a 25-bit mask, the subnet spans IP addresses `192.168.2.0 - 192.168.2.127`.

GPFS does not impose limits on the number of bits in the subnet mask.

This feature cannot be used to establish fault tolerance or automatic failover. If the interface corresponding to an IP address in the list is down, GPFS does not use the next one on the list.

When you use subnets, both the interface corresponding to the daemon address and the interface that matches the subnet settings must be operational.

For more information about subnets, see *Using remote access with public and private IP addresses* in the *IBM Spectrum Scale: Administration Guide*.

Specifying a cluster name or a cluster name pattern for each subnet is needed only when a private network is shared across clusters. If the use of a private network is confined within the local cluster, then you do not need to specify the cluster name in the subnet specification.

Limitation and fix: Although there is no upper limit to the number of subnets that can be specified in the `subnets` option, a limit does exist as to the number of subnets that are listed in the `subnets` option that a given node can be a part of. That limit is seven for nodes that do not have a fix that increases the limit and 64 for nodes that do have the fix. For example, the 7-subnet limit precludes the effective use of more than seven network interfaces on a node if each interface belongs to a distinct subnet that is listed in the `subnets` option.

The fix that increases the limit to 64 is included as part of the following APARs: `IJ06771` for Version 4.1.1, `IJ06770` for 4.2.3, and `IJ06762` for 5.0.1.

If a node exceeds the limit, then some or all of its network interfaces that belong to the subnets in the `subnets` option might not be used in communicating with other nodes, with the primary GPFS daemon interface being used instead.

sudoUser={*UserName* | DELETE}
Specifies a non-root admin user ID to be used when sudo wrappers are enabled and a root-level background process calls an administration command directly instead of through **sudo**. The GPFS daemon that processes the administration command specifies this non-root user ID instead of the root ID when it needs to run internal commands on other nodes. For more information, see the topic *Root-level processes that call administration commands directly* in the *IBM Spectrum Scale: Administration Guide*.

UserName
Enables this feature and specifies the non-root admin user ID.

DELETE
Disables this feature, as in the following example:

```
mmchconfig sudoUser=DELETE
```

syncBuffsPerIteration
This parameter is used to expedite buffer flush and the rename operations that are done by MapReduce jobs.

The default value is 100. It should be set to 1 for the GPFS FPO cluster for Big Data applications. Keep it as the default value for all other cases.

syncSambaMetadataOps
Is used to enable and disable the syncing of metadata operations that are issued by the SMB server.

If set to **yes**, **fsync()** is used after each metadata operation to provide reasonable failover behavior on node failure. This ensures that the node taking over can see the metadata changes. Enabling **syncSambaMetadataOps** can affect performance due to more sync operations.

If set to **no**, the additional sync overhead is avoided at the potential risk of losing metadata updates after a failure.

systemLogLevel
Specifies the minimum severity level for messages that are sent to the system log. The severity levels from highest to lowest priority are: `alert`, `critical`, `error`, `warning`, `notice`, `configuration`, `informational`, `detail`, and debug. The value that is specified for this attribute can be any severity level, or the value none can be specified so no messages are sent to the system log. The default value is `notice`.

GPFS generates some critical log messages that are always sent to the system logging service. This attribute only affects messages originating in the GPFS daemon (`mmfsd`). Log messages originating in some administrative commands are only stored in the GPFS log file.

This attribute is only valid for Linux nodes.

tiebreakerDisks
Controls whether GPFS uses the node-quorum-with-tiebreaker algorithm in place of the regular node-based quorum algorithm. See the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*. To enable this feature, specify the names of 1 - 3 disks. Separate the NSD names with a semicolon (;) and enclose the list in quotes. The disks do not have to belong to any particular file system, but must be directly accessible from the quorum nodes. For example:

```
tiebreakerDisks="gpfs1nsd;gpfs2nsd;gpfs3nsd"
```

To disable this feature, use:

```
tiebreakerDisks=no
```

When you change the tiebreaker disks, be aware of the following requirements:

- In a CCR-based cluster, if the disks that are specified in the **tiebreakerDisks** parameter belong to any file system, the GPFS daemon does not need to be up on all of the nodes in the cluster. However, any file system that contains the disks must be available to run the `mmlsfs` command.
- In a traditional server-based (non-CCR) configuration repository cluster, the GPFS daemon must be shut down on all the nodes of the cluster.

Note: When you add or delete a **tiebreakerCheck** event, IBM Spectrum Scale must be down on all the nodes of the cluster. For more information, see “mmaddcallback command” on page 12 and “mmdelcallback command” on page 362.

tscCmdPortRange=*Min-Max*
Specifies the range of port numbers to be used for extra TCP/IP ports that some administration commands need for their processing. Defining a port range makes it easier for you set firewall rules that allow incoming traffic on only those ports. For more information, see the topic *IBM Spectrum Scale port usage* in the *IBM Spectrum Scale: Administration Guide*.

If you used the **spectrumscale** installation toolkit to install a version of IBM Spectrum Scale that is earlier than version 5.0.0, then this attribute is initialized to 60000-61000. Otherwise, this attribute is initially undefined and the port numbers are dynamically assigned from the range of ephemeral ports that are provided by the operating system.

uidDomain
Specifies the UID domain name for the cluster.

GPFS must be down on all the nodes when changing the `uidDomain` attribute.

See the IBM white paper *UID Mapping for GPFS in a Multi-cluster Environment* (https://www.ibm.com/docs/en/spectrum-scale?topic=STXKQY/uid_gpfs.pdf).

unmountOnDiskFail={yes | no | meta}
Controls how the GPFS daemon responds when it detects a disk failure:

yes
The GPFS daemon force-unmounts the file system that contains the failed disk. Other file systems on the local node and all the nodes in the cluster continue to function normally, unless they have a dependency on the failed disk.

Note: The local node can remount the file system when the disk problem is resolved.

Use this setting in the following situations:

- The cluster contains SAN-attached disks in large multinode configurations and does not use replication.
- A node in the cluster hosts **descOnly** disks. Other nodes in the cluster should not set this value. For more information, see the topic *Data Mirroring and replication* in the *IBM Spectrum Scale: Administration Guide*.

no
This setting is the default. The GPFS daemon marks the disk as failed, notifies all nodes that use the disk that the disk has failed, and continues to function without the failed disk as long it can. If the number of failure groups with a failed disk is the same as or greater than the metadata replication factor, the daemon panics the file system. Note that the metadata replication factor that is used for checking is the current metadata replication set or actual replication factor in effect for log files. If all failure groups contain failed disks, then the daemon panics the file system, instead of marking the disk down.

Note: When the disk problem is resolved, issue the `mmchdisk <file system> start` command to make the disk active again.

This setting is appropriate when the node is using metadata-and-data replication, because the cluster can work from the replica until the failed disk is active again.

meta
This setting has the same effect as **no**, except that the GPFS daemon does not panic the file system unless it cannot access any replica of the metadata. Except that even if all failure groups contain the failed data disks, the daemon still marks the data disks down, instead of panicking the file system as with the **no** option.

However, subsequent data flush attempts on failed data disks might still result in panicking the file system.

Note that the intention of this setting is to allow users to list all directories and read some files, even if some or all data disks are not available for read.

Important: Set the attribute to `meta` for FPO deployment or when the metadata replication factor and the data replication factor are greater than one.

The `-N` flag is valid for this attribute.

usePersistentReserve
Specifies whether to enable or disable Persistent Reserve (PR) on the disks. Valid values are `yes` or `no` (`no` is the default). GPFS must be stopped on all nodes when setting this attribute.

To enable PR and to obtain recovery performance improvements, your cluster requires a specific environment:

- All disks must be PR-capable.
- On AIX, all disks must be hdisks; on Linux, they must be generic (`/dev/sd*`) or DM-MP (`/dev/dm-*`) disks.
- If the disks have defined NSD servers, all NSD server nodes must be running the same operating system (AIX or Linux).
- If the disks are SAN-attached to all nodes, all nodes in the cluster must be running the same operating system (AIX or Linux).

For more information, see *Reduced recovery time using Persistent Reserve* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

verbsHungRDMATimeout
Specifies the number of seconds that IBM Spectrum Scale waits before waking up a thread that is waiting for a response to an RDMA request. The default value is 30 seconds. The valid range is 15 - 8640000 seconds. This feature cleans up long-waiting threads that wait for the completion event of RDMA read, write and send work requests which might never receive a response.

A sample RDMA read log entry is shown:

```
2020-07-28_19:06:49.972-0400: [I] RDMA hang break: wake up thread 28097
(waiting 33.82 sec for RDMA read on index 0 cookie 1)
```

verbsPorts
Specifies the addresses for RDMA transfers between an NSD client and server, where an address can be either of the following identifiers:

- InfiniBand device name, port number, and fabric number
- Network interface name and fabric number

You must enable `verbsRdma` to enable `verbsPorts`. If you want to specify a network interface name and a fabric number, you must enable `verbsRdmaCm`.

The format for `verbsPorts` is as follows:

```
verbsPorts="{ibAddress | niAddress}[ {ibAddress | niAddress}...]"
```

where:

ibAddress
Is an InfiniBand address with the following format:

```
Device[/Port[/Fabric]]
```

where:

Device
Is the HCA device name.

Port
Is a one-based port number, such as 1 or 2. The default value is 1. If you do not specify a port, then the port is 1.

Fabric
Is the number of an InfiniBand (IB) fabric (IB subnet on a switch). The default value is 0. If you do not specify a fabric number, then the fabric number is 0.

niAddress
Is a network interface address with the following format:

```
Interface[/Fabric]
```

where:

Interface
Is a network interface name.

Fabric
Is the number of an InfiniBand (IB) fabric (IB subnet on a switch). The default value is 0. If you do not specify a fabric number, then the fabric number is 0.

For this attribute to take effect, you must restart the GPFS daemon on the nodes on which the value of `verbsPorts` changed.

The `-N` flag is valid for this attribute.

The following examples might be helpful:

- The following assignment creates two RDMA connections between an NSD client and server that use both ports of a dual-ported adapter with fabric number 7 on port 1 and fabric number 8 on port 2:

```
verbsPorts="mlx4_0/1/7 mlx4_0/2/8"
```

- The following assignment, without the fabric number, creates two RDMA connections between an NSD client and server that use both ports of a dual-ported adapter with the fabric number defaulting to 0:

```
verbsPorts="mthca0/1 mthca0/2"
```

- The following assignment creates two RDMA connections between an NSD client and server that use network interface names. The first connection is network interface ib3 with a default fabric number of 0. The second connection is network interface ib2 with a fabric number of 7:

```
verbsPorts="ib3 ib2/7"
```

- The following assignment creates four RDMA connections that include both InfiniBand addresses and network interface addresses:

```
verbsPorts="ib3 ib2/7 mlx4_0/1/7 mlx4_0/2/8"
```

verbsPortsWaitTimeout
Specifies the number of seconds that the GPFS daemon startup service waits for the RDMA ports on a node to become active. The default value is 60 seconds. If the timeout expires, the startup service takes the action that is specified by the attribute `verbsRdmaFailBackTCPIfNotAvailable`. This attribute applies only to the RDMA ports that are specified by the `verbsPorts` attribute. For more information, see the topic *Suboptimal performance due to VERBS RDMA being inactive* in the *IBM Spectrum Scale: Problem Determination Guide*.

Note: To monitor the state of the RDMA ports, the GPFS startup service requires the following commands to be installed on the node:

```
/usr/sbin/ibstat
/usr/bin/ibdev2netdev
/usr/bin/netstat
```

The `-N` flag is valid for this attribute. This attribute takes effect when IBM Spectrum Scale is restarted.

verbsRdma
Enables or disables InfiniBand RDMA using the Verbs API for data transfers between an NSD client and NSD server. Valid values are `enable` or `disable`.

The `-N` flag is valid for this attribute.

verbsRdmaCm
Enables or disables the RDMA Connection Manager (RDMA CM or RDMA_CM) using the RDMA_CM API for establishing connections between an NSD client and NSD server. Valid values are `enable` or `disable`. You must enable `verbsRdma` to enable `verbsRdmaCm`.

If RDMA CM is enabled for a node, the node is only able to establish RDMA connections using RDMA CM to other nodes with `verbsRdmaCm` enabled. RDMA CM enablement requires IPoIB (IP over InfiniBand) with an active IP address for each port. Although IPv6 must be enabled, the GPFS implementation of RDMA CM does not currently support IPv6 addresses, so an IPv4 address must be used.

If `verbsRdmaCm` is not enabled when `verbsRdma` is enabled, the older method of RDMA connection prevails.

The `-N` flag is valid for this attribute.

verbsRdmaFailBackTCPIfNotAvailable={yes | no}
Specifies the action for the GPFS startup service to take if the timeout that is specified by the `verbsPortsWaitTimeout` attribute expires. This attribute applies only to the RDMA ports that are specified by the `verbsPorts` attribute.

Note: To monitor the state of the RDMA ports, the GPFS startup service requires the following commands to be installed on the node:

```
/usr/sbin/ibstat
/usr/bin/ibdev2netdev
/usr/bin/netstat
```

yes
The GPFS startup service configures communication for the GPFS daemon based on the number of active RDMA ports:

- If some of the RDMA ports are active, the GPFS daemon is configured to use the active RDMA ports for RDMA transfers.
- If none of the RDMA ports are active, the GPFS daemon is configured to use the TCP/IP connections of the node for RDMA transfers.

no
The startup service exits, even if some of the RDMA ports are active. Correct the problem and restart IBM Spectrum Scale by running the `mmstartup` command on the node.

The `-N` flag is valid for this attribute. This attribute takes effect when IBM Spectrum Scale is restarted.

verbsRdmaPkey
Specifies an InfiniBand partition key for a connection between the specified node and an Infiniband server that is included in an InfiniBand partition. This parameter is valid only if `verbsRdmaCm` is set to `disable`.

Only one partition key is supported per IBM Spectrum Scale cluster.

The `-N` flag is valid for this attribute.

verbsRdmaRoCEToS

Specifies the Type of Service (ToS) value for clusters using RDMA over Converged Ethernet (RoCE). Acceptable values for this parameter are 0, 8, 16, and 24. The default value is -1.

If the user-specified value is neither the default nor an acceptable value, the script exits with an error message to indicate that no change has been made. However, a RoCE cluster continues to operate with an internally set ToS value of 0 even if the `mmchconfig` command failed. Different ToS values can be set for different nodes or groups of nodes.

The `-N` flag is valid for this attribute.

The `verbsPorts` parameter can use IP netmask/subnet to specify network interfaces to use for RDMA CM. However, this format is allowed only when `verbsRdmaCm=yes`. Otherwise these entries are ignored. This allows the use of VLANs and multiple IP interfaces per IB device in general.

verbsRdmaSend
Enables or disables the use of InfiniBand RDMA send and receive rather than TCP for most GPFS daemon-to-daemon communication. Valid values are `yes` or `no`. The default value is `no`. The `verbsRdma` option must be enabled and valid `verbsPorts` must be defined before `verbsRdmaSend` can be enabled.

When the attribute is set to `no`, only data transfers between an NSD server and an NSD client are eligible for RDMA. When the attribute is set to `yes`, the GPFS daemon uses InfiniBand RDMA connections for daemon-to-daemon communications only with nodes that are at IBM Spectrum Scale 5.0.0 or later.

If verbsRdmaSend is enabled, then the value set for the nsdInlineWriteMax parameter is ignored. In such cases, you can set the maximum transaction size that can be sent as embedded data in an NSD-write RPC by using the verbsRecvBufferSize parameter.

The -N flag is valid for this attribute.

verbsRecvBufferCount
Defines the number of RDMA recv buffers created for each RDMA connection that is enabled for RDMA send when verbsRdmaSend is enabled. The default value is 128.

The -N flag is valid for this attribute.

verbsRecvBufferSize
Defines the size, in bytes, of the RDMA send and recv buffers that are used for RDMA connections that are enabled for RDMA send when verbsRdmaSend is enabled. This parameter also specifies the maximum transaction size that can be sent as embedded data in an NSD-write RPC. The default value is 4096.

The -N flag is valid for this attribute.

workerThreads
Controls an integrated group of variables that tune file system performance. Use this variable to tune file systems in environments that are capable of high sequential or random read/write workloads or small-file activity. For new installations of the product, this variable is preferred over worker1Threads and preFetchThreads.

The default value is 48. If protocols are installed, then the default value is 512. The valid range is 1-8192. However, the maximum value of workerThreads plus preFetchThreads plus nsdMaxWorkerThreads is 8192. The -N flag is valid with this variable.

This variable controls both internal and external variables. The internal variables include maximum settings for concurrent file operations, for concurrent threads that flush dirty data and metadata, and for concurrent threads that prefetch data and metadata. You can further adjust the external variables with the mmchconfig command:

```
logBufferCount
prefetchThreads
worker3Threads
```

The prefetchThreads parameter is described in this help topic. See the Tuning Parameters article in the IBM Spectrum Scale wiki in developerWorks® for descriptions of the logBufferCount and worker3Threads parameters.

Important: After you set workerThreads to a non-default value, avoid setting worker1Threads. If you do, at first only worker1Threads is changed. But when IBM Spectrum Scale is restarted, all corresponding variables are automatically tuned according to the value of worker1Threads, instead of workerThreads.

worker1Threads
For some categories of file I/O, this variable controls the maximum number of concurrent file I/O operations. You can increase this value to increase the I/O performance of the file system. However, increasing this variable beyond some point might begin to degrade file system performance.

Important: After you set workerThreads to a non-default value, avoid setting worker1Threads. If you do, at first only worker1Threads is changed. But when IBM Spectrum Scale is restarted, all corresponding variables are automatically tuned according to the value of worker1Threads, instead of workerThreads.

This attribute is primarily used for random read or write requests that cannot be pre-fetched, random I/O requests, or small file activity. The default value is 48. The minimum value is 1. The maximum value of prefetchThreads plus worker1Threads plus nsdMaxWorkerThreads is 8192 on all 64-bit platforms.

The -N flag is valid for this attribute.

writebehindThreshold

The `writebehindThreshold` parameter specifies the point at which GPFS starts flushing new data out of the page pool for a file that is being written sequentially. Until the file size reaches this threshold, no write-behind is started because the full blocks are filled.

Increasing this value defers write-behind for new larger files, which can be useful. The workload folder contains temporary files that are smaller than the value of `writebehindThreshold` and are deleted before they are flushed from cache. The default value of this parameter is 512 KiB. If the value is too large, there might be too many dirty buffers that the sync thread has to flush at the next sync interval, causing a surge in disk IO. Keeping the value small ensures a smooth flow of dirty data to disk.

Note: If you set new values for **afmParallelReadChunkSize**, **afmParallelReadThreshold**, **afmParallelWriteChunkSize**, and **afmParallelWriteThreshold**; you need not relink filesets for the new values to take effect.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmchconfig` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

To change the maximum file system block size that is allowed to 8 MiB, issue the following command:

```
# mmchconfig maxblocksize=8M
```

A sample output is as follows:

```
Verifying GPFS is stopped on all nodes ...
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

To confirm the change, issue the following command:

```
# mmlsconfig
```

A sample output is as follows:

```
Configuration data for cluster ib.cluster:
------------------------------------------
clusterName ib.cluster
clusterId 13882433899463047326
autoload no
minReleaseLevel 5.1.1.x
dmapiFileHandleSize 32
maxblocksize 8M
pagepool 2g
[c21f1n18]
pagepool 5g
[common]
verbsPorts mthca0/1
verbsRdma enable
```

```
subnets 10.168.80.0
adminMode central

File systems in cluster ib.cluster:
-----------------------------------
/dev/fs1
```

See also

- “mmaddnode command” on page 34
- “mmchnode command” on page 244
- “mmcrcluster command” on page 306
- “mmdelnode command” on page 375
- “mmlsconfig command” on page 495
- “mmlscluster command” on page 492

Location

/usr/lpp/mmfs/bin

mmchdisk command

Changes state or parameters of one or more disks in a GPFS file system.

Synopsis

```
mmchdisk Device {resume | start} -a
           [-N {Node[,Node...] | NodeFile | NodeClass}]
           [-o InodeResultFile][--inode-criteria CriteriaFile]
           [--pit-continues-on-error] [--qos QosClass]
```

or

```
mmchdisk Device {suspend | empty | resume | stop | start | change}
           {-d "DiskDesc[;DiskDesc...]" | -F StanzaFile}
           [-N {Node[,Node...] | NodeFile | NodeClass}]
           [-o InodeResultFile][--inode-criteria CriteriaFile]
           [--pit-continues-on-error] [--qos QosClass]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the mmchdisk command to change the state or the parameters of one or more disks in a GPFS file system.

The state of a disk is a combination of its status and availability, displayed with the mmlsdisk command. Disk status is normally either ready, emptied, suspended, or to be emptied. A transitional status such as replacing, replacement, or being emptied might appear if a disk is being deleted or replaced. An emptied disk indicates that there is not any file system data or metadata on the disk and that new data will not be put on the disk. A disk status of emptied means that the disk can be removed without needing to migrate data. A suspended or being emptied disk is one that the user has decided not to place any new data on. Existing data on a suspended or being emptied disk may still be read or updated. Typically, a disk is suspended before it is removed from a file system. When a disk is suspended, the mmrestripefs command migrates the data off that disk. Disk availability is either up or down.

Be sure to use stop before you take a disk offline for maintenance. You should also use stop when a disk has become temporarily inaccessible due to a disk failure that is repairable without loss of data on that disk (for example, an adapter failure or a failure of the disk electronics).

The *Disk Usage* (dataAndMetadata, dataOnly, metadataOnly, or descOnly) and *Failure Group* parameters of a disk are adjusted with the change option. See the *Recoverability considerations* topic in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

The mmchdisk change command does not move data or metadata that resides on the disk. After you issue the mmchdisk change command for *DiskUsage* or *State*, you might have to issue the mmrestripefs command with the -r option to relocate data so that it conforms to the new disk parameters.

The mmchdisk command can be issued for a mounted or unmounted file system. When maintenance is complete or the failure has been repaired, use the mmchdisk command with the start option. If the failure cannot be repaired without loss of data, you can use the mmdeldisk command.

Note: The mmchdisk command does not change the NSD servers that are associated with the disk, or the storage pool of the disk:

- To change the NSD servers use the mmchnsd command.
- To change the storage pool use the mmdeldisk and mmadddisk commands.

Prior to GPFS 3.5, the disk information for the mmchdisk change option was specified in the form of disk descriptors defined as follows (with the second, third, sixth and seventh fields reserved):

```
DiskName:::DiskUsage:FailureGroup:::
```

For backward compatibility, the mmchdisk command still accepts the traditional disk descriptor format, but its use is deprecated.

Parameters

Device
The device name of the file system to which the disks belong. File system names need not be fully-qualified. fs0 is as acceptable as /dev/fs0.

This must be the first parameter.

suspend
or
empty
Instructs GPFS to stop allocating space on the specified disk. Put a disk in this state when you are preparing to remove the file system data from the disk or if you want to prevent new data from being put on the disk. This is a user-initiated state that GPFS never enters without an explicit command to change the disk state. Existing data on a suspended disk may still be read or updated.

A disk remains in a suspended or to be emptied state until it is explicitly resumed. Restarting GPFS or rebooting nodes does not restore normal access to a suspended disk.

resume
Informs GPFS that a disk previously suspended is now available for allocating new space. If the disk is currently in a stopped state, it remains stopped until you specify the start option. Otherwise, normal read and write access to the disk resumes.

stop
Instructs GPFS to stop any attempts to access the specified disks. Use this option to tell the file system manager that a disk has failed or is currently inaccessible because of maintenance.

A disk remains stopped until it is explicitly started by the mmchdisk command with the start option.

You cannot run mmchdisk stop on a file system with a default replication setting of 1.

start
Informs GPFS that disks previously stopped are now accessible. This is accomplished by first changing the disk availability from down to recovering. The file system metadata is then scanned and any missing updates (replicated data that was changed while the disk was down) are repaired. If this operation is successful, the availability is changed to up. If the metadata scan fails, availability is set to unrecovered. This situation can occur if too many disks in the file system are down. The metadata scan can be re-initiated at a later time by issuing the mmchdisk start command again.

If more than one disk in the file system is down, they must all be started at the same time by issuing the mmchdisk *Device* start -a command. If you start them separately and metadata is stored on any disk that remains down, the mmchdisk start command fails.

change
Instructs GPFS to change the disk usage parameter, the failure group parameter, or both, according to the values specified in the NSD stanzas.

-d "*DiskDesc*[;*DiskDesc*...]"
A descriptor for each disk to be changed.

Specify only disk names when using the suspend, resume, stop, or start options. Delimit multiple disk names with semicolons and enclose the list in quotation marks. For example, "gpfs1nsd;gpfs2nsd"

When using the `change` option, include the disk name and any new *Disk Usage* and *Failure Group* positional parameter values in the descriptor. Delimit descriptors with semicolons and enclose the list in quotation marks; for example, `"gpfs1nsd:::dataOnly;gpfs2nsd:::metadataOnly:12"`.

The use of disk descriptors is discouraged.

-F *StanzaFile*
Specifies a file containing the NSD stanzas for the disks to be changed. NSD stanzas have the following format:

```
%nsd:
  nsd=NsdName
  usage={dataOnly | metadataOnly | dataAndMetadata | descOnly}
  failureGroup=FailureGroup
  pool=StoragePool
  servers=ServerList
  device=DiskName
  thinDiskType={no | nvme | scsi | auto}
```

where:

nsd=*NsdName*
The name of the NSD to change. For a list of disks that belong to a particular file system, issue the `mmlsnsd -f` *Device*, `mmlsfs` *Device* `-d`, or `mmlsdisk` *Device* command. The `mmlsdisk` *Device* command will also show the current disk usage and failure group values for each of the disks. This clause is mandatory for the `mmchdisk` command.

usage={dataOnly | metadataOnly | dataAndMetadata | descOnly}
Specifies the type of data to be stored on the disk:

dataAndMetadata
Indicates that the disk contains both data and metadata. This is the default for disks in the system pool.

dataOnly
Indicates that the disk contains data and does not contain metadata. This is the default for disks in storage pools other than the system pool.

metadataOnly
Indicates that the disk contains metadata and does not contain data.

descOnly
Indicates that the disk contains no data and no file metadata. IBM Spectrum Scale uses this type of disk primarily to keep a copy of the file system descriptor. It can also be used as a third failure group in certain disaster recovery configurations. For more information, see the topic *Synchronous mirroring utilizing GPFS replication* in the *IBM Spectrum Scale: Administration Guide*.

This clause is meaningful only for the `mmchdisk change` option.

failureGroup=*FailureGroup*
Identifies the failure group to which the disk belongs. A failure group identifier can be a simple integer or a topology vector that consists of up to three comma-separated integers. The default is -1, which indicates that the disk has no point of failure in common with any other disk.

GPFS uses this information during data and metadata placement to ensure that no two replicas of the same block can become unavailable due to a single failure. All disks that are attached to the same NSD server or adapter must be placed in the same failure group.

If the file system is configured with data replication, all storage pools must have two failure groups to maintain proper protection of the data. Similarly, if metadata replication is in effect, the system storage pool must have two failure groups.

Disks that belong to storage pools in which write affinity is enabled can use topology vectors to identify failure domains in a shared-nothing cluster. Disks that belong to traditional storage pools must use simple integers to specify the failure group.

pool=*StoragePool*
Specifies the storage pool to which the disk is to be assigned. If this name is not provided, the default is `system`.

Only the system storage pool can contain `metadataOnly`, `dataAndMetadata`, or `descOnly` disks. Disks in other storage pools must be `dataOnly`.

servers=*ServerList*
A comma-separated list of NSD server nodes. This clause is ignored by the `mmadddisk` command.

device=*DiskName*
The block device name of the underlying disk device. This clause is ignored by the `mmadddisk` command.

thinDiskType={no | nvme | scsi | auto}
Specifies the space reclaim disk type:

no
The disk does not support space reclaim. This value is the default.

nvme
The disk is a TRIM capable NVMe device that supports the `mmreclaimspace` command.

scsi
The disk is a thin provisioned SCSI disk that supports the `mmreclaimspace` command.

auto
The type of the disk is either `nvme` or `scsi`. IBM Spectrum Scale will try to detect the actual disk type automatically. To avoid problems, you should replace `auto` with the correct disk type, `nvme` or `scsi`, as soon as you can.

Note: The space reclaim auto-detection is enhanced in IBM Spectrum Scale 5.0.5. Use the auto keyword after you upgrade the cluster to IBM Spectrum Scale 5.0.5 or later.

-a
Specifies to change the state of all of the disks belonging to the file system, *Device*. This operand is valid only on the `resume` and `start` options.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass* }
Specifies a list of nodes that should be used for making the requested disk changes. This command supports all defined node classes. The default is `all` or the current value of the `defaultHelperNodes` parameter of the `mmchconfig` command.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

--inode-criteria *CriteriaFile*
Specifies the interesting inode criteria flag, where *CriteriaFile* contains a list of the following flags with one per line:

BROKEN
Indicates that a file has a data block with all of its replicas on disks that have been removed.

Note: BROKEN is always included in the list of flags even if it is not specified.

dataUpdateMiss
Indicates that at least one data block was not updated successfully on all replicas.

exposed
Indicates an inode with an exposed risk; that is, the file has data where all replicas are on suspended disks. This could cause data to be lost if the suspended disks have failed or been removed.

illCompressed
Indicates an inode in which file compression or decompression is deferred, or in which a compressed file is partly decompressed to allow the file to be written into or memory-mapped.

illPlaced
Indicates an inode with some data blocks that might be stored in an incorrect storage pool.

illReplicated
Indicates that the file has a data block that does not meet the setting for the replica.

metaUpdateMiss
Indicates that there is at least one metadata block that has not been successfully updated to all replicas.

unbalanced
Indicates that the file has a data block that is not well balanced across all the disks in all failure groups.

Note: If a file matches *any* of the specified interesting flags, all of its interesting flags (even those not specified) will be displayed.

-o *InodeResultFile*
Contains a list of the inodes that met the interesting inode flags that were specified on the `--inode-criteria` parameter. The output file contains the following:

INODE_NUMBER
This is the inode number.

DISKADDR
Specifies a dummy address for later `tsfindinode` use.

SNAPSHOT_ID
This is the snapshot ID.

ISGLOBAL_SNAPSHOT
Indicates whether or not the inode is in a global snapshot. Files in the live file system are considered to be in a global snapshot.

INDEPENDENT_FSETID
Indicates the independent fileset to which the inode belongs.

MEMO (*INODE_FLAGS FILE_TYPE* [*ERROR*])
Indicates the inode flag and file type that will be printed:

`Inode flags`:

```
BROKEN
exposed
dataUpdateMiss
illCompressed
illPlaced
illReplicated
metaUpdateMiss
unbalanced
```

`File types`:

```
BLK_DEV
CHAR_DEV
DIRECTORY
FIFO
LINK
LOGFILE
REGULAR_FILE
RESERVED
SOCK
*UNLINKED*
*DELETED*
```

Notes:

1. An error message will be printed in the output file if an error is encountered when repairing the inode.
2. `DISKADDR`, `ISGLOBAL_SNAPSHOT`, and `FSET_ID` work with the `tsfindinode` tool (`/usr/lpp/mmfs/bin/tsfindinode`) to find the file name for each inode. `tsfindinode` uses the output file to retrieve the file name for each interesting inode.

--pit-continues-on-error
Allows the **mmchdisk** command to continue repairing the remaining files, if errors are encountered in the parallel inode traverse (PIT) phase that performs scanning of user file metadata. An output file is generated if an error occurs in the PIT phase. The location of the file that logs the errors is displayed in the command output.

Note:

- The **mmchdisk** command continues to run only if the error is reported in the PIT phase of the command execution. If the error is reported in other phases of command execution, the command stops running.
- The `--pit-continues-on-error` option works only if the minimum release level of the IBM Spectrum Scale cluster is 5.1.1 or later.

--qos ***QOSClass***
Specifies the Quality of Service for I/O operations (QoS) class to which the instance of the command is assigned. If you do not specify this parameter, the instance of the command is assigned by default to the `other` QoS class. (Unlike other commands that have the `--qos` option, the `mmchdisk` command runs in the `other` class by default.) This parameter has no effect unless the QoS service is enabled. For more information, see the topic "mmchqos command" on page 263. Specify one of the following QoS classes:

maintenance
This QoS class is typically configured to have a smaller share of file system IOPS. Use this class for I/O-intensive, potentially long-running GPFS commands, so that they contribute less to reducing overall file system performance.

other
This QoS class is typically configured to have a larger share of file system IOPS. Use this class for administration commands that are not I/O-intensive.

For more information, see the topic *Setting the Quality of Service for I/O operations (QoS)* in the *IBM Spectrum Scale: Administration Guide*.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmchdisk` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To `suspend` active disk **gpfs2nsd**, issue the following command:

```
# mmchdisk fs0 suspend -d gpfs2nsd
```

   To confirm the change, issue the following command:

```
# mmlsdisk fs0
```

In IBM Spectrum Scale versions earlier than 4.1.1, the product displays information similar to the following example:

```
disk      driver sector failure holds   holds status        storage
name      type    size  group  metadata data      availability pool
--------- ------ ------ ------- ------- ---- ----- --------- ------
gpfs2nsd   nsd      512     2   yes   yes   suspended  up   system
hd3vsdn01  nsd      512     2   yes   yes   ready      up   system
hd27n01    nsd      512     8   yes   yes   ready      up   system
hd28n01    nsd      512     8   yes   yes   ready      up   system
hd29n01    nsd      512     8   yes   yes   ready      up   system
hd10vsdn09 nsd      512  4003   no    yes   ready      up   sp1
hd11vsdn10 nsd      512  4003   no    yes   ready      up   sp1
```

Note: In product versions earlier than V4.1.1, the `mmlsdisk` command lists the disk status as `suspended`. In product versions V4.1.1 and later, the `mmlsdisk` command lists the disk status as `to be emptied` with both `mmchdisk suspend` or `mmchdisk empty` commands.

2. To `empty` active disk **gpfs1nsd**, issue the following command:

```
# mmchdisk fs0 empty -d gpfs1nsd
```

To confirm the change, issue the following command:

```
# mmlsdisk fs0 -L
```

In product version V4.1.1 and later, the system displays information similar to the following example:

```
disk     driver sector failure holds   holds                                storage
name     type   size   group  metadata data  status     availability  disk id pool
-------- ------ ------ ------ -----    -----  -----      ------------- ------- -----
gpfs1nsd nsd     512      -1  Yes      Yes   to be emptied up              1  system
gpfs2nsd nsd     512      -1  Yes      Yes   to be emptied up              2  system
gpfs3nsd nsd     512      -1  Yes      Yes   ready         up              3  system
gpfs4nsd nsd     512      -1  Yes      Yes   ready         up              4  system
Number of quorum disks: 3
Read quorum value:      2
Write quorum value:     2
Attention: Due to an earlier configuration change the file system
may contain data that is at risk of being lost.
```

3. To specify that metadata should no longer be stored on disk **gpfs1nsd**, issue the following command:

```
# mmchdisk fs0 change -d "gpfs1nsd:::dataOnly"
```

To confirm the change, issue the following command:

```
# mmlsdisk fs0
```

A sample output is as follows:

```
disk      driver  sector failure holds    holds status              storage
name      type     size   group  metadata data         availability pool
--------- ------ ------  ------- ------   ----  -----  ------------ -------
hd2vsdn01  nsd     512      2    yes      yes   ready    up         system
hd3vsdn01  nsd     512      2    yes      yes   ready    up         system
hd27n01    nsd     512      8    yes      yes   ready    up         system
gpfs1nsd   nsd     512      8    no       yes   ready    up         system
hd29n01    nsd     512      8    yes      yes   ready    up         system
hd10vsdn09 nsd     512   4003    no       yes   ready    up         sp1
hd11vsdn10 nsd     512   4003    no       yes   ready    up         sp1
```

4. To start a disk and check for files matching the interesting inode criteria located on the disk, issue the following command:

```
# mmchdisk fs1 start -d vmip2_nsd3 /tmp/crit --inode-criteria
```

A sample output is as follows:

```
mmnsddiscover:  Attempting to rediscover the disks.  This may take a while ...
mmnsddiscover:  Finished.
vmip2.gpfs.net:  GPFS: 6027-1805 [N] Rediscovered nsd server access to vmip2_nsd3.
```

```
GPFS: 6027-589 Scanning file system metadata, phase 1 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 2 ...
Scanning file system metadata for data storage pool
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 3 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 4 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-565 Scanning user file metadata ...
 100.00 % complete on Wed Apr 15 10:20:37 2015  65792 inodes with total  398 MB data
processed)
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-3312 No inode was found matching the criteria.
```

The disk was started successfully. No files matching the requested criteria were found.

See also

- *Displaying GPFS disk states* in the *IBM Spectrum Scale: Administration Guide*.
- “mmadddisk command” on page 28
- “mmchnsd command” on page 254
- “mmdeldisk command” on page 364
- “mmlsdisk command” on page 497
- “mmlsnsd command” on page 522
- “mmrpldisk command” on page 690
- “mmrestripefs command” on page 682

Location

`/usr/lpp/mmfs/bin`

mmcheckquota command

Checks file system user, group and fileset quotas.

Synopsis

```
mmcheckquota [-v] [-N {Node[,Node...] | NodeFile | NodeClass}]
             [--qos QosClass] {-a | Device [:Fileset] [Device [:Fileset] ...]}
```

or

```
mmcheckquota {-u UserQuotaFile | -g GroupQuotaFile | -j FilesetQuotaFile}
             [--qos QosClass] Device
```

or

```
mmcheckquota --backup backupDir Device
```

Availability

Available on all IBM Spectrum Scale editions. Available on AIX and Linux.

Description

The mmcheckquota command serves two purposes:

1. Count inode and space usage in a file system by user, group and fileset, and write the collected data into quota files.

   Note: In cases where small files do not have an additional block allocated for them, quota usage may show less space usage than expected.

2. Replace either the user, group, or fileset quota files, for the file system designated by *Device*, thereby restoring the quota files for the file system. These files must be contained in the root directory of *Device*. If a backup copy does not exist, an empty file is created when the mmcheckquota command is issued.

The mmcheckquota command counts inode and space usage for a file system and writes the collected data into quota files. Indications leading you to the conclusion you should run the mmcheckquota command include:

- MMFS_QUOTA error log entries. This error log entry is created when the quota manager has a problem reading or writing the quota file.
- Quota information is lost due to a node failure. A node failure could leave users unable to open files or deny them disk space that their quotas should allow.
- The in-doubt value is approaching the quota limit.

  The sum of the in-doubt value and the current usage may not exceed the hard limit. Consequently, the actual block space and number of files available to the user of the group may be constrained by the in-doubt value. If the in-doubt value approaches a significant percentage of the quota, use the mmcheckquota command to account for the lost space and files.
- User, group, or fileset quota files are corrupted.

The mmcheckquota command is I/O-intensive and should be run when the system load is light. When issuing the mmcheckquota command on a mounted file system, negative in-doubt values may be reported if the quota server processes a combination of up-to-date and back-level information. This is a transient situation and can be ignored.

If a file system is ill-replicated, the mmcheckquota command will not be able to determine exactly how many valid replicas actually exist for some of the blocks. If this happens, the used block count results

from mmcheckquota will not be accurate. It is recommended that you run mmcheckquota to restore accurate usage count after the file system is no longer ill-replicated.

Parameters

-a
Checks all GPFS file systems in the cluster from which the command is issued.

--backup *BackupDirectory*
Specifies a backup directory, which must be in the same GPFS file system as the root directory of *Device*.

In IBM Spectrum Scale 4.1.1 and later, you can use this parameter to copy quota files. The command copies three quota files to the specified directory.

Device
Specifies the device name of the file system. File system names do not need to be fully-qualified. **fs0** is as acceptable as **/dev/fs0**.

Fileset
Specifies the name of a fileset that is located on a device for which inode and space usage to be counted.

-g *GroupQuotaFileName*
Replaces the current group quota file with the file indicated.

When replacing quota files with the -g option, the quota file must be in the root directory of the GPFS file system.

-j *FilesetQuotaFilename*
Replaces the current fileset quota file with the file indicated.

When replacing quota files with the -j option, the quota file must be in the root directory of the GPFS file system.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies the nodes that will participate in a parallel quota check of the system. This command supports all defined node classes. The default is all or the current value of the defaultHelperNodes parameter of the mmchconfig command.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

-u *UserQuotaFilename*
Replaces the current user quota file with the file indicated.

When replacing quota files with the -u option, the quota file must be in the root directory of the GPFS file system.

--qos *QosClass*
Specifies the Quality of Service for I/O operations (QoS) class to which the instance of the command is assigned. If you do not specify this parameter, the instance of the command is assigned by default to the maintenance QoS class. This parameter has no effect unless the QoS service is enabled. For more information, see the topic “mmchqos command” on page 263. Specify one of the following QoS classes:

maintenance
This QoS class is typically configured to have a smaller share of file system IOPS. Use this class for I/O-intensive, potentially long-running GPFS commands, so that they contribute less to reducing overall file system performance.

other
This QoS class is typically configured to have a larger share of file system IOPS. Use this class for administration commands that are not I/O-intensive.

For more information, see the topic *Setting the Quality of Service for I/O operations (QoS)* in *IBM Spectrum Scale: Administration Guide*.

Options

-v
Reports discrepancies between calculated and recorded disk quotas.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmcheckquota` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

GPFS must be running on the node from which the `mmcheckquota` command is issued.

Examples

1. To check quotas for file system `fs0`, issue the following command:

```
# mmcheckquota fs0
```

   The system displays information only if a problem is found.

2. To check quotas for all file systems, issue the following command:

```
# mmcheckquota -a
```

   The system displays information only if a problem is found or if quota management is not enabled for a file system:

```
fs2: no quota management installed
fs3: no quota management installed
```

3. To report discrepancies between calculated and recorded disk quotas, issue the following command:

```
# mmcheckquota -v fs1
```

   A sample output is as follows:

```
fs1: Start quota check
1 % complete on Fri Apr 17 13:07:47 2009
6 % complete on Fri Apr 17 13:07:48 2009
11 % complete on Fri Apr 17 13:07:49 2009
17 % complete on Fri Apr 17 13:07:50 2009
22 % complete on Fri Apr 17 13:07:51 2009
28 % complete on Fri Apr 17 13:07:52 2009
33 % complete on Fri Apr 17 13:07:53 2009
38 % complete on Fri Apr 17 13:07:54 2009
44 % complete on Fri Apr 17 13:07:55 2009
49 % complete on Fri Apr 17 13:07:56 2009
55 % complete on Fri Apr 17 13:07:57 2009
61 % complete on Fri Apr 17 13:07:58 2009
66 % complete on Fri Apr 17 13:07:59 2009
72 % complete on Fri Apr 17 13:08:00 2009
78 % complete on Fri Apr 17 13:08:01 2009
```

```
83 % complete on Fri Apr 17 13:08:02 2009
89 % complete on Fri Apr 17 13:08:03 2009
94 % complete on Fri Apr 17 13:08:04 2009
Finished scanning the inodes for fs1.
Merging results from scan.
fs1: quota check found the following differences:
USR 0: 288400 subblocks counted (was 288466); 24 inodes counted (was 81)
USR 60011: 50 subblocks counted (was 33); 2 inodes counted (was 20)
USR 60012: 225 subblocks counted (was 223); 9 inodes counted (was 4)
USR 60013: 175 subblocks counted (was 146); 7 inodes counted (was 26)
USR 60014: 200 subblocks counted (was 178); 8 inodes counted (was 22)
USR 60015: 275 subblocks counted (was 269); 11 inodes counted (was 0)
USR 60019: 0 subblocks counted (was 9); 0 inodes counted (was 5)
USR 60020: 0 subblocks counted (was 1); 0 inodes counted (was 3)
GRP 0: 28845098 subblocks counted (was 28844639); 14 inodes counted (was 91)
FILESET 0: 28849125 subblocks counted (was 28848717); 105 inodes counted (was 24)
```

See also

- “mmedquota command” on page 404
- “mmfsck command” on page 410
- “mmlsquota command” on page 535
- “mmquotaon command” on page 654
- “mmquotaoff command” on page 651
- “mmrepquota command” on page 666

Location

/usr/lpp/mmfs/bin

mmchfileset command

Changes the attributes of a GPFS fileset.

Synopsis

```
mmchfileset Device {FilesetName | -J JunctionPath}
              [-j NewFilesetName] [-t NewComment] [-p afmAttribute=Value...]
              [--allow-permission-change PermissionChangeMode]
              [--inode-limit MaxNumInodes[:NumInodesToPreallocate]]
              [--iam-mode Mode]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmchfileset` command changes attributes for an existing GPFS fileset.

For information on GPFS filesets, see the *IBM Spectrum Scale: Administration Guide*.

Parameters

Device
The device name of the file system that contains the fileset.

File system names need not be fully qualified. `fs0` is as acceptable as `/dev/fs0`.

FilesetName
Specifies the name of the fileset.

-J *JunctionPath*
Specifies the junction path name for the fileset.

A junction is a special directory entry that connects a name in a directory of one fileset to the root directory of another fileset.

-j *NewFilesetName*
Specifies the new name that is to be given to the fileset. This name must be fewer than 256 characters in length. The root fileset cannot be renamed.

-t *NewComment*
Specifies an optional comment that appears in the output of the `mmlsfileset` command. This comment must be fewer than 256 characters in length. This option cannot be used on the root fileset.

-p afmAttribute=*Value*
Specifies an AFM configuration attribute and its value. More than one `-p` option can be specified.

The following AFM configuration attributes are valid:

afmAsyncDelay
Specifies (in seconds) the amount of time by which write operations are delayed (because write operations are asynchronous with respect to remote clusters). For write-intensive applications that keep writing to the same set of files, this delay is helpful because it replaces multiple writes to the home cluster with a single write containing the latest data. However, setting a very high value weakens the consistency of data on the remote cluster.

This configuration parameter is applicable only for writer caches (Single writer, Independent writer, and Primary) in which data from cache is pushed to home.

Valid values are 1 - 2147483647. The default is 15.

afmDirLookupRefreshInterval
Controls the frequency of data revalidations that are triggered by such lookup operations as `ls` or `stat` (specified in seconds). When a lookup operation is done on a directory, if the specified amount of time has passed, AFM sends a message to the home cluster to find out whether the metadata of that directory has been modified since the last time it was checked. If the time interval has not passed, AFM does not check the home cluster for updates to the metadata.

Valid values are 0 through 2147483647. The default is 60. In situations where home cluster data changes frequently, a value of 0 is recommended.

afmDirOpenRefreshInterval
Controls the frequency of data revalidations that are triggered by such I/O operations as `read` or `write` (specified in seconds). After a directory has been cached, open requests resulting from I/O operations on that object are directed to the cached directory until the specified amount of time has passed. Once the specified amount of time has passed, the open request gets directed to a gateway node rather than to the cached directory.

Valid values are between 0 and 2147483647. The default is 60. Setting a lower value guarantees a higher level of consistency.

afmEnableAutoEviction
Enables eviction on a given fileset. A `yes` value specifies that eviction is allowed on the fileset. A `no` value specifies that eviction is not allowed on the fileset.

See also the topic about cache eviction in the *IBM Spectrum Scale: Administration Guide*.

afmExpirationTimeout
Is used with `afmDisconnectTimeout` (which can be set only through `mmchconfig`) to control how long a network outage between the cache and home clusters can continue before the data in the cache is considered out of sync with home. After `afmDisconnectTimeout` expires, cached data remains available until `afmExpirationTimeout` expires, at which point the cached data is considered expired and cannot be read until a reconnect occurs.

Valid values are 0 through 2147483647. The default is disable.

afmFastCreate
Enable at the AFM cache and AFM-DR primary fileset level. AFM sends RPC to the gateway node for each update that is happening on the fileset. If the workload mostly involves new files creation, this parameter reduces the RPC exchanges between the application and the gateway node, improves the application performance, and minimizes the memory queue requirement at the gateway node.

To set or unset the **`afmFastCreate`** parameter on an AFM or AFM-DR fileset, you need to stop or unlink the fileset. For more information, see *Stop and start replication on a fileset* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Valid values are 'yes' or no'.

afmFileLookupRefreshInterval
Controls the frequency of data revalidations that are triggered by such lookup operations as `ls` or `stat` (specified in seconds). When a lookup operation is done on a file, if the specified amount of time has passed, AFM sends a message to the home cluster to find out whether the metadata of the file has been modified since the last time it was checked. If the time interval has not passed, AFM does not check the home cluster for updates to the metadata.

Valid values are 0 through 2147483647. The default is 30. In situations where home cluster data changes frequently, a value of 0 is recommended.

afmFileOpenRefreshInterval
Controls the frequency of data revalidations that are triggered by such I/O operations as `read` or `write` (specified in seconds). After a file has been cached, open requests resulting from I/O operations on that object are directed to the cached file until the specified amount of time has passed. Once the specified amount of time has passed, the open request gets directed to a gateway node rather than to the cached file.

Valid values are 0 through 2147483647. The default is 30. Setting a lower value guarantees a higher level of consistency.

afmGateway
Specifies the user-defined gateway node for an AFM or AFM DR fileset, that gets preference over internal hashing algorithm. If the specified gateway node is not available, then AFM internally assigns a gateway node from the available list to the fileset. **afmHashVersion** value must be already set as '5'. Before setting the **afmGateway** value; stop replication on the fileset or unlink the AFM fileset. After setting the value, start replication on the fileset or link the AFM fileset.

An example of setting the gateway node by using stop and start AFM fileset -

```
#mmafmctl <fs> stop -j <fileset>
#mmchfileset <fs> <fileset> -p afmGateway=<GateWayNode>
#mmlsfileset <fs>  -L --afm
#mmafmctl <fs> start -j <fileset>
```

An example of setting the gateway node by using unlink and link AFM fileset -

```
#mmunlinkfileset <fs> <fileset> -f
#mmchfileset <fs> <fileset> -p afmGateway=<GateWayNode>
#mmlsfileset <fs>  -L --afm
#mmlinkfileset <fs> -J <filesetPath>
```

To revert to the internal hashing algorithm of assigning gateway nodes, you must delete the assigned gateway node value of the AFM fileset.

An example of deleting the gateway node by using stop and start AFM fileset -

```
#mmafmctl <fs> stop -j <fileset>
#mmchfileset <fs> <fileset> -p afmGateway=delete
#mmlsfileset <fs>  -L --afm
#mmafmctl <fs> start -j <fileset>
```

An example of deleting the gateway node by using unlink and link AFM fileset -

```
#mmunlinkfileset <fs> <fileset> -f
#mmchfileset <fs> <fileset> -p afmGateway=delete
#mmlsfileset <fs>  -L --afm
#mmlinkfileset <fs> -J <filesetPath>
```

Note: Ensure that the file system is upgraded to IBM Spectrum Scale 5.0.2 or later.

This parameter also accepts 'all' as a value. **afmGateway**=*all* can be set on RO and LU mode AFM fileset. This parameter supports the value 'all' for the file system-level migration. This value improves the file system level migration performance by using the inode-based hashing. When this value is set, all operations that belong to an inode are queued to the gateway node. It helps load balance operations by using inode number.

Set **afmGateway**=*all* by using the stop and start AFM fileset. For example,

```
# mmafmctl <fs> stop -j <fileset>
# mmchfileset <fs> <fileset> -p afmGateway=all
# mmafmctl <fs> start -j <fileset>
```

afmMode
Specifies the mode in which the cache operates. Valid values are the following:

single-writer | sw
Specifies single-writer mode.

read-only | ro
Specifies read-only mode. (For `mmcrfileset`, this is the default value.)

local-updates | lu
Specifies local-updates mode.

independent-writer | iw
Specifies independent-writer mode.

Primary | drp
Specifies the primary mode for AFM asynchronous data replication.

Secondary | drs
Specifies the secondary mode for AFM asynchronous data replication.

Changing from single-writer/read-only modes to read-only/local-updates/single-writer is supported. When changing from read-only to single-writer, the read-only cache is up-to-date. When changing from single-writer to read-only, all requests from cache should have been played at home. Changing from local-updates to read-only/local-updates/single-writer is restricted. A typical dataset is set up to include a single cache cluster in single-writer mode (which generates the data) and one or more cache clusters in local-updates or read-only mode. AFM single-writer/independent-writer filesets can be converted to primary. Primary/secondary filesets cannot be converted to AFM filesets.

In case of AFM asynchronous data replication, the `mmchfileset` command cannot be used to convert to primary from secondary. For detailed information, see *AFM-based Asynchronous Disaster Recovery (AFM DR)* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

For more information, see the topic about caching modes in the *IBM Spectrum Scale: Administration Guide* chapter about active file management.

afmNumFlushThreads
Defines the number of threads used on each gateway to synchronize updates to the home cluster. The default value is 4, which is sufficient for most installations. The current maximum value is 1024, which is too high for most installations; setting this parameter to such an extreme value should be avoided.

afmNumReadThreads
Defines the number of threads that can be used on each participating gateway node during parallel read. The default value of this parameter is 1; that is, one reader thread will be active on every gateway node for each big read operation qualifying for splitting per the parallel read threshold value. The valid range of values is 1 to 64.

afmNumWriteThreads
Defines the number of threads that can be used on each participating gateway node during parallel write. The default value of this parameter is 1; that is, one writer thread will be active on every gateway node for each big write operation qualifying for splitting per the parallel write threshold value. Valid values can range from 1 to 64.

afmParallelMounts

When this parameter is enabled, the primary gateway node of a fileset at a cache cluster attempts to mount the exported path from multiple NFS servers that are defined in the mapping. Then, this primary gateway node sends unique messages through each NFS mount to improve performance by transferring data in parallel.

Before enabling this parameter, define the mapping between the primary gateway node and NFS servers by issuing the **`mmafmconfig`** command.

afmParallelReadChunkSize
Defines the minimum chunk size of the read that needs to be distributed among the gateway nodes during parallel reads. Values are interpreted in terms of bytes. The default value of this parameter is 128 MiB, and the valid range of values is 0 to 2147483647. It can be changed cluster wide with the `mmchconfig` command. It can be set at fileset level using `mmcrfileset` or `mmchfileset` commands.

afmParallelReadThreshold
Defines the threshold beyond which parallel reads become effective. Reads are split into chunks when file size exceeds this threshold value. Values are interpreted in terms of MiB. The default

value is 1024 MiB. The valid range of values is 0 to 2147483647. It can be changed cluster wide with the `mmchconfig` command. It can be set at fileset level using `mmcrfileset` or `mmchfileset` commands.

afmParallelWriteChunkSize
Defines the minimum chunk size of the write that needs to be distributed among the gateway nodes during parallel writes. Values are interpreted in terms of bytes. The default value of this parameter is 128 MiB, and the valid range of values is 0 to 2147483647. It can be changed cluster wide with the `mmchconfig` command. It can be set at fileset level using `mmcrfileset` or `mmchfileset` commands.

afmParallelWriteThreshold
Defines the threshold beyond which parallel writes become effective. Writes are split into chunks when file size exceeds this threshold value. Values are interpreted in terms of MiB. The default value of this parameter is 1024 MiB, and the valid range of values is 0 to 2147483647. It can be changed cluster wide with the `mmchconfig` command. It can be set at fileset level using `mmcrfileset` or `mmchfileset` commands.

afmPrefetchThreshold
Controls partial file caching and prefetching. Valid values are the following:

0
Enables full file prefetching. This is useful for sequentially accessed files that are read in their entirety, such as image files, home directories, and development environments. The file will be prefetched after three blocks have been read into the cache.

1-99
Specifies the percentage of file size that must be cached before the entire file is prefetched. A large value is suitable for a file accessed either randomly or sequentially but partially, for which it might be useful to ingest the rest of the file when most of it has been accessed.

100
Disables full file prefetching. This value only fetches and caches data that is read by the application. This is useful for large random-access files, such as databases, that are either too big to fit in the cache or are never expected to be read in their entirety. When all data blocks are accessed in the cache, the file is marked as cached.

0 is the default value.

For local-updates mode, the whole file is prefetched when the first update is made.

afmPrimaryId
Specifies the unique primary ID of the primary fileset for asynchronous data replication. This is used for connecting a secondary to a primary.

afmReadDirOnce
Enables AFM to perform one-time `readdir` of a directory from the home after the data migration to the cache and the application is started on the cache data. That is, the application is moved from the home to the cache and the application modifies the directory, which makes the directory dirty. When this parameter is set for a fileset, the prefetch operation is run on the fileset by using `--readdir-only` to move new or modified data from the home to the cache, even if the cache directory is dirty. After the data migration to the cache and the application is started on the cache data, this parameter synchronizes new files at the home directory to the cache for a single time.

Valid values are 'yes' and 'no'. This parameter can be set on AFM RO, LU, and IW filesets. This parameter is not useful for the DMAPI migration.

afmReadSparseThreshold
Specifies the size in MB for files in cache beyond which sparseness is maintained. For all files below the specified threshold, sparseness is not maintained.

afmRefreshOnce
Enables AFM to perform revalidation on files and directories only one time. This parameter improves the application performance after the data migration to the cache and the application is started on the cache data during the migration from old system to the new system. When this

parameter is set to yes, files and directories revalidation is performed only one time. Therefore, only one revalidation request goes to the home or target.

Valid values are 'yes' and 'no'. This parameter can be set on AFM RO, LU, and IW filesets. This parameter is not useful for the DMAPI migration.

afmRPO
Specifies the recovery point objective (RPO) interval for an AFM DR fileset. This attribute is disabled by default. You can specify a value with the suffix `M` for minutes, `H` for hours, or `W` for weeks. For example, for 12 hours specify 12H. If you do not add a suffix, the value is assumed to be in minutes. The range of valid values is 720 minutes - 2147483647 minutes.

afmShowHomeSnapshot
Controls the visibility of the home snapshot directory in cache. For this to be visible in cache, this variable has to be set to `yes`, and the snapshot directory name in the cache and home cannot be the same.

yes
Specifies that the home snapshot link directory is visible.

no
Specifies that the home snapshot link directory is not visible.

For more information about the snapshot, see *Peer snapshot -psnap* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

afmTarget
The only allowed value is `disable`. It is used to convert AFM filesets to regular independent filesets; for example:

```
mmchfileset fs1 ro -p afmTarget=disable
```

After an AFM fileset is converted to a regular fileset, the fileset cannot be changed back to an AFM fileset.

--allow-permission-change *PermissionChangeMode*
Specifies the new permission change mode. This mode controls how **chmod** and ACL operations are handled on objects in the fileset. Valid modes are as follows:

chmodOnly
Specifies that only the UNIX change mode operation (**chmod**) is allowed to change access permissions (ACL commands and API will not be accepted).

setAclOnly
Specifies that permissions can be changed using ACL commands and API only (**chmod** will not be accepted).

chmodAndSetAcl
Specifies that **chmod** and ACL operations are permitted. If the **chmod** command (or **setattr** file operation) is issued, the result depends on the type of ACL that was previously controlling access to the object:

- If the object had a Posix ACL, it will be modified accordingly.
- If the object had an NFSv4 ACL, it will be replaced by the given UNIX mode bits.

Note: This is the default setting when a fileset is created.

chmodAndUpdateAcl
Specifies that **chmod** and ACL operations are permitted. If **chmod** is issued, the ACL will be updated by privileges derived from UNIX mode bits.

--inode-limit *MaxNumInodes*[:*NumInodesToPreallocate*]
Specifies the new inode limit for the inode space that is owned by the specified fileset. The *FilesetName* or *JunctionPath* must refer to an independent fileset. The *NumInodesToPreallocate* specifies an optional number of additional inodes to pre-allocate for the inode space. Use the `mmchfs` command to change inode limits for the root fileset.

The *MaxNumInodes* and *NumInodesToPreallocate* values can be specified with a suffix, for example 100 K or 2 M.

Note: Preallocated inodes cannot be deleted or moved to another independent fileset. It is recommended to avoid preallocating too many inodes because there can be both performance and memory allocation costs associated with such preallocations. In most cases, there is no need to preallocate inodes because GPFS dynamically allocates inodes as needed.

--iam-mode *Mode*
Specifies an integrated archive manager (IAM) mode for the fileset. You can set IAM modes to modify some of the file-operation restrictions that normally apply to immutable filesets. The IAM modes are as follows, listed in order of increasing strictness:

```
ad | advisory
nc | noncompliant
co | compliant
cp | compliant-plus
```

Note that IAM modes can be upgraded from **advisory** to **noncompliant** to **compliant** to **compliant-plus**, but not downgraded. When a fileset is set to one of these IAM modes, a number of operations on files and directories are no longer allowed. Because the IAM mode for a fileset cannot be downgraded, those disallowed operations become persistent and cannot be undone. For more information, see the topic *Immutability and appendOnly features* in the *IBM Spectrum Scale: Administration Guide*.

Note: If you set new values for **afmParallelReadChunkSize**, **afmParallelReadThreshold**, **afmParallelWriteChunkSize**, and **afmParallelWriteThreshold**; you need not relink filesets for the new values to take effect.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority or be a fileset owner to run the `mmchfileset` command with the `-t` option. All other options require root authority.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. The following command renames fileset `fset1` to `fset2` and gives it the comment "first fileset":

   ```
   mmchfileset gpfs1 fset1 -j fset2 -t 'first fileset'
   ```

   The command displays the following information:

   ```
   Fileset 'fset1' changed.
   ```

2. To confirm the change, issue the following command:

   ```
   mmlsfileset gpfs1 -L
   ```

   The system displays the following information:

```
Filesets in file system 'gpfs1':
Name Id RootInode ParentId Created                  InodeSpace MaxInodes AllocInodes Comment
root  0         3       -- Mon Apr 12 16:31:05 2010    0         8001536     8001536 root
fileset
fset2 2     13569        0 Mon Apr 12 16:32:28 2010    0               0           0 first
fileset
```

See also

- “mmchfs command” on page 232
- “mmcrfileset command” on page 311
- “mmdelfileset command” on page 369
- “mmlinkfileset command” on page 485
- “mmlsfileset command” on page 501
- “mmunlinkfileset command” on page 735

Location

/usr/lpp/mmfs/bin

mmchfs command

Changes the attributes of a GPFS file system.

Synopsis

```
mmchfs Device [-A {yes | no | automount}] [-D {posix | nfs4}] [-E {yes | no}]
       [-k {posix | nfs4 | all}] [-K {no | whenpossible | always}]
       [-L LogFileSize] [-m DefaultMetadataReplicas] [-n NumNodes]
       [-o MountOptions] [-r DefaultDataReplicas] [-S {yes | no | relatime}]
       [-T Mountpoint] [-t DriveLetter] [-V {full | compat}] [-z {yes | no}]
       [--filesetdf | --nofilesetdf]
       [--inode-limit MaxNumInodes[:NumInodesToPreallocate]]
       [--log-replicas LogReplicas] [--mount-priority Priority]
       [--perfileset-quota | --noperfileset-quota]
       [--rapid-repair | --norapid-repair]
       [--write-cache-threshold HAWCThreshold]
```

```
mmchfs Device -Q {yes | no}
```

or

```
mmchfs  Device [--maintenance-mode] {yes [--wait] | no }
```

or

```
mmchfs Device -W NewDeviceName
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmchfs` command to change the attributes of a GPFS file system.

Parameters

Device
The device name of the file system to be changed.

File system names need not be fully qualified. `fs0` is as acceptable as `/dev/fs0`. However, file system names must be unique across GPFS clusters.

This must be the first parameter.

-A {yes | no | automount}
Indicates when the file system is to be mounted:

yes
When the GPFS daemon starts.

no
The file system is mounted manually.

automount
On non-Windows nodes, when the file system is first accessed. On Windows nodes, when the GPFS daemon starts.

Note:

- The file system must be unmounted before the automount settings are changed.
- IBM Spectrum Protect for Space Management does not support file systems with the `-A` option set to automount.

-D {nfs4 | posix}
Specifies whether a deny-write open lock blocks writes, which is required by NFS V4, Samba, and Windows. File systems that support NFS V4 must have `-D nfs4` set. The option `-D posix` allows NFS writes even in the presence of a deny-write open lock. If you intend to export the file system on NFS V4 or Samba, or mount your file system on Windows, you must use `-D nfs4`. For NFS V3 (or if the file system is not NFS exported at all) use `-D posix`.

-E {yes | no}
Specifies whether to report exact `mtime` values. If `-E no` is specified, the `mtime` value is periodically updated. If you want to always display exact modification times, specify `-E yes`.

Important: The new value takes effect the next time the file system is mounted.

-k {posix | nfs4 | all}
Specifies the type of authorization supported by the file system:

posix
Traditional GPFS ACLs only (NFS V4 and Windows ACLs are not allowed). Authorization controls are unchanged from earlier releases.

nfs4
Support for NFS V4 and Windows ACLs only. Users are not allowed to assign traditional GPFS ACLs to any file system objects (directories and individual files).

all
Any supported ACL type is permitted. This includes traditional GPFS (`posix`) and NFS V4 and Windows ACLs (`nfs4`).

The administrator is allowing a mixture of ACL types. For example, `fileA` may have a `posix` ACL, while `fileB` in the same file system may have an NFS V4 ACL, implying different access characteristics for each file depending on the ACL type that is currently assigned.

Avoid specifying `nfs4` or `all` unless files are exported to NFS V4 or Samba clients, or the file system is mounted on Windows. NFS V4 and Windows ACLs affect file attributes (mode) and have access and authorization characteristics that are different from traditional GPFS ACLs.

-K {no | whenpossible | always}
Specifies whether strict replication is to be enforced:

no
Strict replication is not enforced. GPFS tries to create the needed number of replicas, but still returns EOK if it can allocate at least one replica.

whenpossible
Strict replication is enforced provided the disk configuration allows it. If there is only one failure group, strict replication is not enforced.

always
Strict replication is enforced.

For more information, see the topic *Strict replication* in the *IBM Spectrum Scale: Problem Determination Guide*.

-L *LogFileSize*
Specifies the size of the internal log files. The *LogFileSize* must be a multiple of the metadata block size. The default log file size is 32 MiB in most cases. However, if the data block size (parameter **-B**) is less than 512 KiB or if the metadata block size (parameter **--metadata-block-size**) is less than 256 KiB, then the default log file size is either 4 MiB or the metadata block size, whichever is greater. The minimum size is 256 KiB and the maximum size is 1024 MiB. Specify this value with the K or M character, for example: 8M. For more information, see “mmcrfs command” on page 318.

The default log size works well in most cases. An increased log file size is useful when the highly available write cache feature (parameter `--write-cache-threshold`) is enabled.

The new log file size is not effective until you apply one of the two following methods:

- The first method requires you in part to restart the GPFS daemon on the manager nodes, but you can do so one node at a time. Follow these steps:
  1. Restart the GPFS daemon (mmfsd) on all the manager nodes of the local cluster. This action is required even if the affected file system is not mounted on any of the manager nodes. You can do this action one manager node at a time.
  2. Remount the file system on all the local and remote nodes that have it mounted. You can do this action one node at a time. The new log file size becomes effective when the file system is remounted on the last affected node.
- The second method requires you to unmount the file system on all the affected nodes at the same time. Follow these steps:
  1. Unmount the file system on all local and remote nodes that have it mounted. The file system must be in the unmounted state on all the nodes at the same time.
  2. Remount the file system on any or all the affected nodes.

-m *DefaultMetaDataReplicas*
Changes the default number of metadata replicas. Valid values are 1, 2, and 3. This value cannot be greater than the value of *MaxMetaDataReplicas* set when the file system was created.

Changing the default replication settings using the `mmchfs` command does not change the replication setting of existing files. After running the `mmchfs` command, the `mmrestripefs` command with the `-R` option can be used to change *all* existing files or you can use the `mmchattr` command to change a small number of existing files.

-n *NumNodes*
Changes the number of nodes for a file system but does not change the existing system metadata structures. This setting is just an estimate and can be used only to affect the layout of the system metadata for storage pools created after the setting is changed.

-o *MountOptions*
Specifies the mount options to pass to the mount command when mounting the file system. For a detailed description of the available mount options, see *GPFS-specific mount options* in the *IBM Spectrum Scale: Administration Guide*.

-Q {yes | no}
If `-Q yes` is specified, quotas are activated automatically when the file system is mounted. If `-Q no` is specified, the quota files remain in the file system, but are not used.

For more information, see the topic *Enabling and disabling GPFS quota management* in the *IBM Spectrum Scale: Administration Guide*.

-r *DefaultDataReplicas*
Changes the default number of data replicas. Valid values are 1, 2, and 3. This value cannot be greater than the value of *MaxDataReplicas* set when the file system was created.

Changing the default replication settings using the **mmchfs** command does not change the replication setting of existing files. After running the `mmchfs` command, the `mmrestripefs` command with the **-R** option can be used to change *all* existing files or you can use the `mmchattr` command to change a small number of existing files.

-S {yes | no | relatime}
Controls how the file attribute **atime** is updated.

Note: The attribute **atime** is updated locally in memory, but the value is not visible to other nodes until after the file is closed. To get an accurate value of **atime**, an application must call subroutine **gpfs_stat** or **gpfs_fstat**.

yes
The **atime** attribute is not updated. The subroutines **gpfs_stat** and **gpfs_fstat** return the time that the file system was last mounted with `relatime=no`. For more information, see the topics "mmmount command" on page 545 with the **-o** parameter and *Mount options specific to IBM Spectrum Scale* in the *IBM Spectrum Scale: Administration Guide*.

no
The **atime** attribute is updated whenever the file is read. This option is the default if the minimum release level (**minReleaseLevel**) of the cluster is less than 5.0.0 when the file system is created.

relatime
The **atime** attribute is updated whenever the file is read, but only if one of the following conditions is true:

- The current file access time (**atime**) is earlier than the file modification time (**mtime**).
- The current file access time (**atime**) is greater than the **atimeDeferredSeconds** attribute. For more information, see the topic *mmchconfig command* in the *IBM Spectrum Scale: Command and Programming Reference*.

This setting is the default if the minimum release level (**minReleaseLevel**) of the cluster is 5.0.0 or greater when the file system is created.

For more information, see the topic *atime values* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

-T ***Mountpoint***
Change the mount point of the file system starting at the next mount of the file system.

The file system must be unmounted on all nodes before this command is issued.

-t ***DriveLetter***
Changes the Windows drive letter for the file system.

The file system must be unmounted on all nodes before the command is issued.

-V {full | compat}
Changes the file system format to the latest format supported by the currently installed level of GPFS. This option might cause the file system to become permanently incompatible with earlier releases of GPFS.

Note: The **-V** option cannot be used to make file systems that were created earlier than GPFS 3.2.1.5 available to Windows nodes. Windows nodes can mount only file systems that were created with GPFS 3.2.1.5 or later.

Before issuing **-V**, see *Migration, coexistence and compatibility* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*. Ensure that all nodes in the cluster have been updated to the latest level of GPFS code and that you have successfully run the `mmchconfig release=LATEST` command.

For information about specific file system format and function changes when you upgrade to the current release, see the topic *File system format changes between versions of GPFS* in the *IBM Spectrum Scale: Administration Guide*.

full
Enables all new functionality that requires different on-disk data structures. Nodes in remote clusters that are running an earlier version of IBM Spectrum Scale will no longer be able to mount the file system. With this option the command fails if it is issued while any node that has the file system mounted is running an earlier version of IBM Spectrum Scale.

compat
Enables only backward-compatible format changes. Nodes in remote clusters that were able to mount the file system before the format changes can continue to do so afterward.

-W ***NewDeviceName***
Assign *NewDeviceName* to be the device name for the file system.

-z {yes | no}
Enable or disable DMAPI on the file system. Turning this option on requires an external data management application such as IBM Spectrum Protect hierarchical storage management (HSM) before the file system can be mounted.

Note: IBM Spectrum Protect for Space Management does not support file systems with the `-A` option set to automount.

For further information regarding DMAPI for GPFS, see *GPFS-specific DMAPI events* in the *IBM Spectrum Scale: Command and Programming Reference*.

--filesetdf
Specifies that when quotas are enforced for a fileset (other than the root fileset), the `df` command reports either quota limit and usage or inode space capacity and usage for the fileset and not for the total file system. This option affects the `df` command behavior only on Linux nodes.

The `df` command reports quota limit and quota usage if quota is enabled for the fileset. If quota is disabled and `filesetdf` is enabled in IBM Spectrum Scale 5.1.1 or later with file system version 5.1.1 or later, then the **df** command reports inode space capacity and inode usage at the independent fileset-level. However, the **df** command reports the block space at the file system-level because the block space is shared with the whole file system.

Note: If quota is enabled, then no behavior change for **df** command with **filesetdf** enabled, regardless of the cluster and file system versions.

--nofilesetdf
Specifies that the numbers reported by the `df` command are not based on the quotas for a fileset. The `df` command returns the numbers for the entire file system. This is the default.

--inode-limit *MaxNumInodes*[:*NumInodesToPreallocate*]
MaxNumInodes specifies the maximum number of files that can be created. Allowable values range from the current number of created inodes (determined by issuing the `mmdf` command with `-F`), through the maximum number of files possibly supported as constrained by the formula:

maximum number of files = (total file system space) / (inode size + subblock size)

Note: This formula works only for simpler configurations. For complex configurations, such as separation of data and metadata, this formula might not provide an accurate result.

If your file system has additional disks added or the number of inodes was insufficiently sized at file system creation, you can change the number of inodes and hence the maximum number of files that can be created.

For file systems that do parallel file creates, if the total number of free inodes is not greater than 5% of the total number of inodes, there is the potential for slowdown in file system access. Take this into consideration when changing your file system.

NumInodesToPreallocate specifies the number of inodes that are preallocated by the system right away. If this number is not specified, GPFS allocates inodes dynamically as needed.

The *MaxNumInodes* and *NumInodesToPreallocate* values can be specified with a suffix, for example 100K or 2M. Note that in order to optimize file system operations, the number of inodes that are actually created may be greater than the specified value.

This option applies only to the root fileset. Preallocated inodes cannot be deleted or moved to another independent fileset. It is recommended to avoid preallocating too many inodes because there can be both performance and memory allocation costs associated with such preallocations. In most cases, there is no need to preallocate inodes because GPFS dynamically allocates inodes as needed. When there are multiple inode spaces, use the `--inode-limit` option of the `mmchfileset` command to alter the inode limits of independent filesets. The `mmchfileset` command can also be used to modify the inode limit of the root inode space. The `--inode-limit` option of the `mmlsfs` command shows the sum of the inode limits of all inode spaces in the file system. Use the `mmlsfileset` command to see the inode limit of the root fileset.

--log-replicas *LogReplicas*
Specifies the number of recovery log replicas. Valid values are 1, 2, 3, or DEFAULT. If DEFAULT is specified, the number of log replicas is the same as the number of metadata replicas currently in effect for the file system and changes when this number is changed.

Changing the default replication settings using the `mmchfs` command does not change the replication setting of existing files. After running the `mmchfs` command, the `mmrestripefs` command with the `-R` option can be used to change existing log files.

This option is applicable only if the recovery log is stored in the `system.log` storage pool. For more information about the `system.log` storage pool, see the topic *The system.log storage pool* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

--maintenance-mode *Device* {yes [--wait] | no}
Turns file system maintenance mode on or off when you change file system attributes. The values are yes and no (the default).

Specifying **`--wait`**, which is valid only when you use it with the yes parameter, turns on the maintenance mode after you have unmounted the file system. If you specify yes without **`--wait`** and the file system is mounted, the command fails.

For more information on maintenance mode, see *File system maintenance mode* in the *IBM Spectrum Scale: Administration Guide*.

--mount-priority *Priority*
Controls the order in which the individual file systems are mounted at daemon startup or when one of the `all` keywords is specified on the `mmmount` command.

File systems with higher *Priority* numbers are mounted after file systems with lower numbers. File systems that do not have mount priorities are mounted last. A value of zero indicates no priority.

--perfileset-quota
Sets the scope of user and group quota limit checks to the individual fileset level, rather than to the entire file system.

`--noperfileset-quota`
Sets the scope of user and group quota limit checks to the entire file system, rather than to individual filesets.

--rapid-repair
Keeps track of incomplete replication on an individual file block basis (as opposed to the entire file). This may result in a faster repair time when very large files are only partially ill-replicated.

--norapid-repair
Specifies that replication status is kept on a whole file basis (rather than on individual block basis).

--write-cache-threshold *HAWCThreshold*
Specifies the maximum length (in bytes) of write requests that will be initially buffered in the highly-available write cache before being written back to primary storage. Only synchronous write requests are guaranteed to be buffered in this fashion.

A value of 0 disables this feature. 64K is the maximum supported value. Specify in multiples of 4K.

This feature can be enabled or disabled at any time (the file system does not need to be unmounted). For more information about this feature, see the topic *Highly-available write cache (HAWC)* in the *IBM Spectrum Scale: Administration Guide*.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmchfs` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

To change the default replicas for metadata to 2 and the default replicas for data to 2 for new files created in the `fs0` file system, issue the following command:

```
# mmchfs fs0 -m 2 -r 2
```

To confirm the change, issue the following command:

```
# mmlsfs fs0 -m -r
```

A sample output is as follows:

```
flag value          description
---- -------------- -----------------------------------
 -m  2              Default number of metadata replicas
 -r  2              Default number of data replicas
```

See also

- “mmchfileset command” on page 224
- “mmcrfs command” on page 318
- “mmdelfs command” on page 373
- “mmdf command” on page 387
- “mmfsck command” on page 410
- “mmlsfs command” on page 506
- “mmrestripefs command” on page 682

Location

`/usr/lpp/mmfs/bin`

mmchlicense command

Controls the type of GPFS license associated with the nodes in the cluster.

Synopsis

```
mmchlicense {client|fpo|server} [--accept] -N {Node[,Node...] | NodeFile | NodeClass}
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmchlicense` command to change the type of GPFS license associated with the nodes in the cluster.

For information on IBM Spectrum Scale license designation, see *IBM Spectrum Scale license designation* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Parameters

client | fpo | server

The type of GPFS license to be assigned to the nodes specified with the `-N` parameter.

client

The IBM Spectrum Scale Client license permits exchange of data between nodes that locally mount the same GPFS file system. No other export of the data is permitted. The GPFS client may not be used for nodes to share GPFS data directly through any application, service, protocol or method, such as Network File System (NFS), Common Internet File System (CIFS), File Transfer Protocol (FTP), or Hypertext Transfer Protocol (HTTP). For these functions, an IBM Spectrum Scale Server license would be required. The use of any of the following components or functions of IBM Spectrum Scale Client is not authorized:

- Configuring a virtual server in the following IBM Spectrum Scale roles: Configuration Manager, Quorum node, Manager node, Network Shared Disk (NSD) Server node, Cluster Export Services node (also known as Protocol node), Advanced File Management (AFM) Gateway node, Transparent Cloud Tiering Gateway node.
- Exporting IBM Spectrum Scale data to virtual servers that do not have a valid IBM Spectrum Scale license through any application, protocol or method, including Network File System (NFS), Server Message Block (SMB), File Transfer Protocol (FTP), Hypertext Transfer Protocol (HTTP), Object Protocol (OpenStack Swift, Amazon S3 API).

server

The IBM Spectrum Scale Server license permits the licensed node to perform GPFS management functions such as cluster configuration manager, quorum node, manager node, and Network Shared Disk (NSD) server. In addition, the IBM Spectrum Scale Server license permits the licensed node to share GPFS data directly through any application, service protocol or method such as NFS, CIFS, FTP, or HTTP. Therefore, protocol nodes also require an IBM Spectrum Scale Server license.

fpo

The IBM Spectrum Scale FPO license permits the licensed node to perform NSD server functions for sharing GPFS data with other nodes that have an IBM Spectrum Scale FPO or IBM Spectrum Scale Server license. This license cannot be used to share data with nodes that have an IBM Spectrum Scale Client license or non-GPFS nodes. The use of any of the following components or functions of IBM Spectrum Scale FPO is not authorized:

- Configuring a virtual server in the following IBM Spectrum Scale roles: Configuration Manager, Quorum node, Manager node, Cluster Export Services node (also known as Protocol node), Advanced File Management (AFM) Gateway node, Transparent Cloud Tiering Gateway node.
- Configuring a virtual server as an IBM Spectrum Scale Network Shared Disk (NSD) Server node for providing IBM Spectrum Scale data access to virtual servers that do not have a valid IBM Spectrum Scale Server or IBM Spectrum Scale FPO license entitlement.
- Exporting IBM Spectrum Scale data to virtual servers that do not have a valid IBM Spectrum Scale license through any application, protocol or method, including Network File System (NFS), Server Message Block (SMB), File Transfer Protocol (FTP), Hypertext Transfer Protocol (HTTP), Object Protocol (OpenStack Swift, Amazon S3 API).

The full text of the Licensing Agreement is provided with the installation media and can be found at the IBM Software license agreements website (www.ibm.com/software/sla/sladb.nsf).

--accept
Indicates that you accept the applicable licensing terms. The license acceptance prompt will be suppressed.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies the nodes that are to be assigned the specified license type.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmchlicense` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

To designate nodes k145n04 and k145n05 as possessing a GPFS server license, issue the following command:

```
# mmchlicense server --accept -N k145n04,k145n05
```

A sample output is as follows:

```
The following nodes will be designated as possessing GPFS server licenses:
        k145n04.kgn.ibm.com
        k145n05.kgn.ibm.com
mmchlicense: Command successfully completed
mmchlicense: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

See also

- “mmlslicense command” on page 511

Location

/usr/lpp/mmfs/bin

mmchmgr command

Assigns a new file system manager node or cluster manager node.

Synopsis

```
mmchmgr {Device | -c} [Node]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmchmgr` command assigns a new file system manager node or cluster manager node.

Parameters

Device
The device name of the file system for which the file system manager node is to be changed. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

-c
Changes the cluster manager node.

Node
The target node to be appointed as either the new cluster manager node or the new file system manager node. Target nodes for manager functions are selected according to the following criteria:

- Target nodes for the cluster manager function must be specified from the list of quorum nodes.
- Target nodes for the file system manager function should be specified from the list of manager nodes, although this is not strictly required.

If *Node* is not specified, the new manager is selected automatically.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmchmgr` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. Assume the file system manager for the file system **gpfs1** is currently **k164n05**. To migrate the file system manager responsibilities to **k164n06**, issue the following command:

```
# mmchmgr gpfs1 k164n06
```

   A sample output is as follows:

```
GPFS: 6027-628 Sending migrate request to current manager node 89.116.68.69 (k164n05).
GPFS: 6027-629 [N] Node 89.116.68.69 (k164n05) resigned as manager for gpfs1.
GPFS: 6027-630 [N] Node 89.116.68.70 (k164n06) appointed as manager for gpfs1.
```

   To verify the change, issue the following command:

```
# mmlsmgr gpfs1
```

   A sample output is as follows:

```
file system manager node [from 89.116.68.69 (k164n06)]
---------------- ------------------
gpfs1 89.116.68.69 (k164n06)
```

2. To change the cluster manager node, issue the following command:

```
# mmchmgr -c c5n107
```

   A sample output is as follows:

```
Appointing node 9.114.132.107 (c5n107) as cluster manager
Node 9.114.132.107 (c5n107) has taken over as cluster manager
```

   To verify the change, issue the following command:

```
# mmlsmgr -c
```

   A sample output is as follows:

```
Cluster manager node: 9.114.132.107 (c5n107)
```

See also

- “mmlsmgr command” on page 515

Location

/usr/lpp/mmfs/bin

mmchnode command

Changes node attributes.

Synopsis

```
mmchnode change-options -N {Node[,Node...] | NodeFile | NodeClass}
     [--cloud-gateway-nodeclass CloudGatewayNodeClass]
```

or

```
mmchnode {-S Filename | --spec-file=Filename}
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmchnode` command to change one or more attributes on a single node or on a set of nodes. If conflicting node designation attributes are specified for a given node, the last value is used. If any of the attributes represent a node-unique value, the **-N** option must resolve to a single node.

Do not use the `mmchnode` command to change the gateway node role while IO is happening on the fileset. Run the `flushpending` command to flush any pending messages from queues before running the `mmchnode` command for the gateway node role changes.

Parameters

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies the nodes whose states are to be changed.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

[--cloud-gateway-nodeclass *CloudGatewayNodeClass*]
Use this option to specify a node class you will use for Transparent cloud tiering management along with the -N option where you will specify individual node names. Both -N with a node list and --cloud-gateway-node with a node class will be required.

-S *Filename* | --spec-file=*Filename*
Specifies a file with a detailed description of the changes to be made. Each line represents the changes to an individual node and has the following format:

```
node-identifier change-options
```

change-options
A blank-separated list of attribute[=*value*] pairs. The following attributes can be specified:

--admin-interface={*hostname* | *ip_address*}
Specifies the name of the node to be used by GPFS administration commands when communicating between nodes. The admin node name must be specified as an IP address or a hostname that is resolved by the host command to the desired IP address. If the keyword DEFAULT is specified, the admin interface for the node is set to be equal to the daemon interface for the node.

--client
Specifies that the node should not be part of the pool of nodes from which cluster managers, file system managers, and token managers are selected.

--cloud-gateway-enable
Enables one or more nodes as Transparent cloud tiering nodes on the cluster based on the -N option parameters.

--cloud-gateway-disable
Disables one or more Transparent cloud tiering nodes from the cluster based on the -N option parameters. Only disable a Transparent cloud tiering node if you no longer need it to migrate or recall data from the configured cloud.

--ces-enable
Enables Cluster Export Services (CES) on the node.

--ces-disable
Disables CES on the node.

--ces-group=*Group*[,*Group*...]
Adds one or more groups to the specified nodes. Each group that is listed is added to all the specified nodes.

--noces-group=*Group*[,*Group*...]
Removes one or more groups from the specified nodes.

--cnfs-disable
Disables the CNFS functionality of a CNFS member node.

--cnfs-enable
Enables a previously-disabled CNFS member node.

--cnfs-groupid=*groupid*
Specifies a failover recovery group for the node. If the keyword DEFAULT is specified, the CNFS recovery group for the node is set to zero.

For more information, see *Implementing a clustered NFS environment on Linux* in *IBM Spectrum Scale: Administration Guide*.

--cnfs-interface=*ip_address_list*
A comma-separated list of host names or IP addresses to be used for GPFS cluster NFS serving.

The specified IP addresses can be real or virtual (aliased). These addresses must be configured to be static (not DHCP) and to not start at boot time.

The GPFS daemon interface for the node cannot be a part of the list of CNFS IP addresses.

If the keyword DEFAULT is specified, the CNFS IP address list is removed and the node is no longer considered a member of CNFS.

If `adminMode` central is in effect for the cluster, all CNFS member nodes must be able to execute remote commands without the need for a password.

For more information, see *Implementing a clustered NFS environment on Linux* in *IBM Spectrum Scale: Administration Guide*.

--daemon-interface={*hostname* | *ip_address*}
Specifies the host name or IP address to be used by the GPFS daemons for node-to-node communication. The host name or IP address must refer to the communication adapter over which the GPFS daemons communicate. Alias interfaces are not allowed. Use the original address or a name that is resolved by the host command to the original address. You cannot set `--daemon-interface=DEFAULT`.

You must stop all the nodes in the cluster with `mmshutdown` before you set this attribute. This requirement holds true even if you are changing only one node.

See examples 8 and 9 at the end of this topic.

If the minimum release level of the cluster is IBM Spectrum Scale 5.1.0 or later, the following features are available:

- You can specify the `--daemon-interface` option for a quorum node even when CCR is enabled. For earlier versions of IBM Spectrum Scale, temporarily change the quorum node to

a nonquorum node, issue the `mmchnode` command with the `--daemon-interface` option for the nonquorum node, and change the nonquorum node back to a quorum node.

- You can change the IP addresses or host names of cluster nodes when a node quorum is not available. For more information, see *Changing IP addresses and host names* in the *IBM Spectrum Scale: Administration Guide*.

--gateway | --nogateway
Specifies whether the node is to be designated as a gateway node or not.

--manager | --nomanager
Designates the node as part of the pool of nodes from which file system managers and token managers are selected.

--nonquorum
Designates the node as a non-quorum node. If two or more quorum nodes are downgraded at the same time, GPFS must be stopped on all nodes in the cluster. GPFS does not have to be stopped if the nodes are downgraded one at a time.

--perfmon | --noperfmon
Specifies whether the node is to be designated as a perfmon node or not.

--nosnmp-agent
Stops the SNMP subagent and specifies that the node should no longer serve as an SNMP collector node. For more information, see *GPFS SNMP support* in *IBM Spectrum Scale: Problem Determination Guide*.

--quorum
Designates the node as a quorum node.

Note: If you are designating a node as a quorum node, and `adminMode central` is in effect for the cluster, you must ensure that GPFS is up and running on that node (`mmgetstate` reports the state of the node as `active`).

--snmp-agent
Designates the node as an SNMP collector node. If the GPFS daemon is active on this node, the SNMP subagent will be started as well. For more information, *GPFS SNMP support* in *IBM Spectrum Scale: Problem Determination Guide*.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmchnode` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To change nodes k145n04 and k145n05 to be both quorum and manager nodes, issue the following command:

```
# mmchnode --quorum --manager -N k145n04,k145n05
```

A sample output is as follows:

```
Wed May 16 04:50:24 EDT 2007: mmchnode: Processing node k145n04.kgn.ibm.com
Wed May 16 04:50:24 EDT 2007: mmchnode: Processing node k145n05.kgn.ibm.com
mmchnode: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

After completion, `mmlscluster` displays information similar to:

```
GPFS cluster information
========================
GPFS cluster name: mynodes.kgn.ibm.com
GPFS cluster id: 680681553700098206
GPFS UID domain: mynodes.kgn.ibm.com
Remote shell command: /usr/bin/ssh
Remote file copy command: /usr/bin/scp

GPFS cluster configuration servers:
-----------------------------------
Primary server: k145n04.kgn.ibm.com
Secondary server: k145n06.kgn.ibm.com

Node Daemon node name  IP address  Admin node name     Designation
-------------------------------------------------------------------
1  k145n04.kgn.ibm.com 9.114.68.68 k145n04.kgn.ibm.com quorum-manager
2  k145n05.kgn.ibm.com 9.114.68.69 k145n05.kgn.ibm.com quorum-manager
3  k145n06.kgn.ibm.com 9.114.68.70 k145n06.kgn.ibm.com
```

2. To change nodes k145n04 and k145n05 to be both quorum and manager nodes, and node k45n06 to be a non-quorum node, issue the following command:

```
# mmchnode -S /tmp/specFile
```

Where the contents of /tmp/specFile are:

```
k145n04 --quorum --manager
k145n05 --quorum --manager
k145n06 --nonquorum
```

A sample output is as follows:

```
Wed May 16 05:23:31 EDT 2007: mmchnode: Processing node k145n04
Wed May 16 05:23:32 EDT 2007: mmchnode: Processing node k145n05
Wed May 16 05:23:32 EDT 2007: mmchnode: Processing node k145n06
Verifying GPFS is stopped on all nodes ...
mmchnode: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

And `mmlscluster` displays information similar to:

```
GPFS cluster information
========================
GPFS cluster name: mynodes.kgn.ibm.com
GPFS cluster id: 680681553700098206
GPFS UID domain: mynodes.kgn.ibm.com
Remote shell command: /usr/bin/rsh
Remote file copy command: /usr/bin/rcp

GPFS cluster configuration servers:
-----------------------------------
Primary server: k145n04.kgn.ibm.com
Secondary server: k145n06.kgn.ibm.com

Node Daemon node name   IP address   Admin node name     Designation
-------------------------------------------------------------------------
1  k145n04.kgn.ibm.com  9.114.68.68  k145n04.kgn.ibm.com quorum-manager
2  k145n05.kgn.ibm.com  9.114.68.69  k145n05.kgn.ibm.com quorum-manager
3  k145n06.kgn.ibm.com  9.114.68.70  k145n06.kgn.ibm.com
```

3. To enable all the nodes specified in the node class TCTNodeClass1 as Transparent cloud tiering nodes, issue the following command:

```
# mmchnode --cloud-gateway-enable -N TCTNodeClass1
```

A sample output is as follows:

```
Wed May 11 12:51:37 EDT 2016: mmchnode: Processing node c350f2u18
mmchnode: Verifying media for Transparent Cloud Tiering nodes...
mmchnode: node c350f2u18 media checks passed.

Wed May 11 12:51:38 EDT 2016: mmchnode: Processing node c350f2u22.pk.labs.ibm.com
mmchnode: node c350f2u22.pok.stglabs.ibm.com media checks passed.

Wed May 11 12:51:41 EDT 2016: mmchnode: Processing node c350f2u26.pk.labs.ibm.com
mmchnode: node c350f2u26.pok.stglabs.ibm.com media checks passed.

mmchnode: Propagating the cluster configuration data to all
affected nodes.  This is an asynchronous process.
```

You can verify the Transparent cloud tiering nodes by issuing the following command:

```
# mmcloudgateway node list
```

4. To designate only a few nodes (node1 and node2) in the node class, *TCTNodeClass1*, as Transparent cloud tiering server nodes, issue the following command:

```
# mmchnode --cloud-gateway-enable -N node1,node2 --cloud-gateway-nodeclass TCTNodeClass1
```

Note: It only designates node1 and node2 as Transparent cloud tiering server nodes from the node class, *TCTNodeClass1*. Administrators can continue to use the node class for other purposes.

5. To disable all Transparent cloud tiering nodes from the node class, TCTNodeClass1, issue the following command:

```
# mmchnode --cloud-gateway-disable -N TCTNodeClass1
```

A sample output is as follows:

```
Thu May 12 16:10:11 EDT 2016: mmchnode: Processing node c350f2u18
mmchnode: Verifying Transparent Cloud Tiering node c350f2u18 can be disabled...
mmchnode: Node c350f2u18 passed disable checks.

Thu May 12 16:10:11 EDT 2016: mmchnode: Processing node c350f2u22.pk.labs.ibm.com
mmchnode: Verifying Transparent Cloud Tiering node c350f2u22.pk.labs.ibm.com can be
disabled...
mmchnode: Node c350f2u22.pok.stglabs.ibm.com passed disable checks.

Thu May 12 16:10:14 EDT 2016: mmchnode: Processing node c350f2u26.pk.labs.ibm.com
mmchnode: Verifying Transparent Cloud Tiering node c350f2u26.pk.labs.ibm.com can be
disabled...
mmchnode: Node c350f2u26.pk.labs.ibm.com passed disable checks.

mmchnode: Propagating the cluster configuration data to all
affected nodes.  This is an asynchronous process.
```

6. To disable only a few nodes (node1 and node2) from the node class, *TCTNodeClass1*, as Transparent cloud tiering server nodes, issue the following command:

```
# mmchnode --cloud-gateway-disable -N node1,node2 --cloud-gateway-nodeclass TCTNodeClass1
```

Note: It only disables node1 and node2 as Transparent cloud tiering server nodes from the node class, *TCTNodeClass1*.

7. To add groups to specified nodes, issue the **mmchnode --ces-group** command. For example:

```
# mmchnode --ces-group g1,g2 -N 2,3
```

Note: This command adds groups g1 and g2 to both nodes 2 and 3. Run the **mmces node list** command to view the group allocation:

```
[root@cluster-12 ~]# mmces node list
Node   Name                       Node Groups   Node Flags
------ ------------------------- ------------- ------------
```

```
2      cluster-12.localnet.com   g1,g2         Suspended
3      cluster-13.localnet.com   g1,g2         none
4      cluster-14.localnet.com                 none
```

8. The following example changes the daemon interface of a cluster node:

   a. The `mmlscluster` command displays the state of the cluster:

```
# mmlscluster
GPFS cluster information
========================
  GPFS cluster name:         small_cluster.localnet.com
  GPFS cluster id:           5072947464461061246
  GPFS UID domain:           small_cluster.localnet.com
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR
GPFS cluster configuration servers:
-----------------------------------
  Primary server:    node-6.localnet.com (not in use)
  Secondary server:  (none)
 Node  Daemon node name     IP address      Admin node name      Designation
--------------------------------------------------------------------------------
   1   node-6.localnet.com  192.168.124.36  node-6.localnet.com  quorum
   2   node-7.localnet.com  192.168.124.37  node-7.localnet.com  quorum
   3   node-8.localnet.com  192.168.124.38  node-8.localnet.com
```

   b. The `mmchnode` command changes the daemon interface of `node-6` from `node-6.localnet.com/192.168.124.36` to `node-6-2.new-localnet.com/10.20.40.36`:

```
# mmchnode --daemon-interface=node-6-2.new-localnet.com -N node-6
Wed Sep 23 20:01:35 CEST 2020: mmchnode: Processing node node-6.localnet.com
Verifying GPFS is stopped on all nodes ...
Wed Sep 23 20:01:36 CEST 2020: mmchnode: Collecting ccr.nodes file version from all
quorum nodes ...
Wed Sep 23 20:01:37 CEST 2020: mmchnode: Applying new change to ccr.nodes version 2 on
all available quorum nodes ...
Wed Sep 23 20:01:40 CEST 2020: mmchnode: Committing new version of ccr.nodes file ...
mmchnode: Propagating the cluster configuration data to all affected nodes.
mmchnode: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

   c. The `mmlscluster` command now shows the new daemon interface for `node-6`. Some lines of output are omitted:

```
# mmlscluster
GPFS cluster information
========================
...
 Node  Daemon node name           IP address      Admin node name      Designation
-------------------------------------------------------------------------------------
   1   node-6-2.new-localnet.com  10.20.40.36     node-6.localnet.com  quorum
   2   node-7.localnet.com        192.168.124.37  node-7.localnet.com  quorum
   3   node-8.localnet.com        192.168.124.38  node-8.localnet.com
```

9. The following example changes the daemon interface and the administration interface of multiple nodes:

   a. The data file `/tmp/specfile` contains the following lines:

```
node-6 --daemon-interface=node-6-2.new-localnet.com --admin-interface=node-6-2.new-
localnet.com
node-7 --daemon-interface=node-7-2.new-localnet.com --admin-interface=node-7-2.new-
localnet.com
node-8 --daemon-interface=node-8-2.new-localnet.com --admin-interface=node-8-2.new-
localnet.com
```

   b. The `mmchmode` command changes the daemon interfaces and the administration interfaces of `node-6`, `node-7`, and `node-8` based on the information in the data file:

```
mmchnode -S /tmp/specFile
Wed Sep 23 20:19:48 CEST 2020: mmchnode: Processing node node-6
Wed Sep 23 20:19:49 CEST 2020: mmchnode: Processing node node-7
Wed Sep 23 20:19:50 CEST 2020: mmchnode: Processing node node-8
Verifying GPFS is stopped on all nodes ...
```

```
Wed Sep 23 20:19:52 CEST 2020: mmchnode: Collecting ccr.nodes file version from all
quorum nodes ...
Wed Sep 23 20:19:53 CEST 2020: mmchnode: Applying new change to ccr.nodes version 4 on
all available quorum nodes ...
Wed Sep 23 20:19:56 CEST 2020: mmchnode: Committing new version of ccr.nodes file ...
mmchnode: Propagating the cluster configuration data to all affected nodes.
mmchnode: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

c. The `mmlscluster` command now shows the new daemon interfaces and administration interfaces for the three nodes. Some lines of output are omitted:

```
# mmlscluster
GPFS cluster information
========================
GPFS cluster name:         small_cluster.localnet.com
...
Node  Daemon node name             IP address      Admin node name
Designation
-----------------------------------------------------------------------------------
   1   node-6-2.new-localnet.com    10.20.40.36    node-6-2.new-localnet.com     quorum
   2   node-7-2.new-localnet.com    10.20.40.37    node-7-2.new-localnet.com     quorum
   3   node-8-2.new-localnet.com    10.20.40.38    node-8-2.new-localnet.com
```

See also

- “mmchconfig command” on page 170
- “mmlscluster command” on page 492

Location

`/usr/lpp/mmfs/bin`

mmchnodeclass command

Changes user-defined node classes.

Synopsis

```
mmchnodeclass ClassName {add | delete | replace}
              -N {Node[,Node...] | NodeFile | NodeClass}
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmchnodeclass` command to make changes to existing user-defined node classes.

Parameters

ClassName
Specifies the name of an existing user-defined node class to modify.

add
Adds the nodes specified with the `-N` option to *ClassName*.

delete
Deletes the nodes specified with the `-N` option from *ClassName*.

replace
Replaces all *ClassName* members with a new list of nodes specified with the `-N` option.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies the member names of nodes and node classes that will be used for the add, `delete`, or `replace` action.

NodeClass cannot be used to add members that already contain other node classes. For example, two user-defined node classes called `siteA` and `siteB` were used to create a new node class called `siteAandB`, as follows:

```
mmcrnodeclass siteAandB -N siteA,siteB
```

The `siteAandB` node class cannot later be specified for *NodeClass* when adding to existing node classes.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmchnodeclass` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

To display the current members of a user-defined node class called `siteA`, issue this command:

```
mmlsnodeclass siteA
```

The system displays information similar to:

```
Node Class Name       Members
--------------------- ------------------------------------------------------------
siteA                 c8f2c4vp1,c8f2c4vp2
```

To add node `c8f2c1vp4` to the member list of the user-defined node class `siteA`, issue this command:

```
mmchnodeclass siteA add -N c8f2c1vp4
```

The system displays information similar to:

```
mmchnodeclass: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

To display the updated member list of `siteA`, issue this command:

```
mmlsnodeclass siteA
```

The system displays information similar to:

```
Node Class Name       Members
--------------------- ------------------------------------------------------------
siteA                 c8f2c1vp4,c8f2c4vp1,c8f2c4vp2
```

To delete node `c8f2c4vp2` from the member list of `siteA`, issue this command:

```
mmchnodeclass siteA delete -N c8f2c4vp2
```

The system displays information similar to:

```
mmchnodeclass: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

To display the updated member list of `siteA`, issue this command:

```
mmlsnodeclass siteA
```

The system displays information similar to:

```
Node Class Name       Members
--------------------- ------------------------------------------------------------
siteA                 c8f2c1vp4,c8f2c4vp1
```

To replace all the current members of `siteA` with the members of node class `linuxNodes`, issue this command:

```
mmchnodeclass siteA replace -N linuxNodes
```

The system displays information similar to:

```
mmchnodeclass: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

To display the updated member list of `siteA`, issue this command:

```
mmlsnodeclass siteA
```

The system displays information similar to:

```
Node Class Name       Members
--------------------- ------------------------------------------------------------
siteA                 linuxNodes
```

See also

- “mmcrnodeclass command” on page 333
- “mmdelnodeclass command” on page 379
- “mmlsnodeclass command” on page 520

Location

/usr/lpp/mmfs/bin

mmchnsd command

Changes Network Shared Disk (NSD) configuration attributes.

Synopsis

```
mmchnsd {"DiskDesc[;DiskDesc...]" | -F StanzaFile}
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The mmchnsd command serves several purposes. You can use it to:

- Specify a server list for an NSD that does not have one.
- Change the NSD server nodes specified in the server list.
- Delete the server list. The disk must now be SAN-attached to all nodes in the cluster on which the file system will be mounted.

In IBM Spectrum Scale 5.0.0 or later, you do not need to unmount the file system before changing NSDs.

Note: This feature applies only to non-vdisk NSDs. To enable this feature, you must upgrade all the nodes in the cluster to IBM Spectrum Scale 5.0.0 or later.

In versions of IBM Spectrum Scale that are earlier than 5.0.0, you must unmount the file system that contains the NSD that is being changed before you issue the mmchnsd command.

You must follow these rules when you change NSDs:

- Identify the disks by the NSD names that were given to them by the mmcrnsd command.
- Explicitly specify values for all NSD servers on the list even if you are only changing one of the values.
- Connect the NSD to the new nodes prior to issuing the mmchnsd command.

Note: The mmchdisk command does not change the name of the NSD, the NSD servers that are associated with the disk, the storage pool of the disk:

- The name of the NSD cannot be changed.
- To change the NSD servers use the mmchnsd command.
- To change the storage pool use the mmdeldisk and mmadddisk commands.

Prior to GPFS 3.5, the disk information was specified in the form of disk descriptors defined as:

```
DiskName:ServerList:
```

For backward compatibility, the mmchnsd command will still accept the traditional disk descriptors but their use is discouraged.

Parameters

DiskDesc
A descriptor for each NSD to be changed. Each descriptor is separated by a semicolon (;). The entire list must be enclosed in single or double quotation marks. The use of disk descriptors is discouraged.

-F *StanzaFile*
Specifies a file containing the NSD stanzas for the disks to be changed. NSD stanzas have this format:

```
%nsd:
  nsd=NsdName
  servers=ServerList
```

```
  usage=DiskUsage
  failureGroup=FailureGroup
  pool=StoragePool
  device=DiskName
  thinDiskType={no | nvme | scsi | auto}
```

where:

nsd=*NsdName*
Is the NSD name that was given to the disk by the `mmcrnsd` command. This clause is mandatory for the `mmchnsd` command.

servers=*ServerList*
Is a comma-separated list of NSD server nodes. You can specify up to eight NSD servers in this list. The defined NSD will preferentially use the first server on the list. If the first server is not available, the NSD will use the next available server on the list.

When specifying server nodes for your NSDs, the output of the `mmlscluster` command lists the host name and IP address combinations recognized by GPFS. The utilization of aliased host names not listed in the `mmlscluster` command output may produce undesired results.

If you do not define a *ServerList*, GPFS assumes that the disk is SAN-attached to all nodes in the cluster. If all nodes in the cluster do not have access to the disk, or if the file system to which the disk belongs is to be accessed by other GPFS clusters, you must specify a value for *ServerList*.

To remove the NSD server list, do not specify a value for *ServerList* (remove or comment out the `servers=`*ServerList* clause of the NSD stanza).

usage=*DiskUsage*
Specifies the type of data to be stored on the disk. If this clause is specified, the value must match the type of usage already in effect for the disk; `mmchnsd` cannot be used to change this value.

failureGroup=*FailureGroup*
Identifies the failure group to which the disk belongs. A failure group identifier can be a simple integer or a topology vector that consists of up to three comma-separated integers. The default is -1, which indicates that the disk has no point of failure in common with any other disk.

GPFS uses this information during data and metadata placement to ensure that no two replicas of the same block can become unavailable due to a single failure. All disks that are attached to the same NSD server or adapter must be placed in the same failure group.

If the file system is configured with data replication, all storage pools must have two failure groups to maintain proper protection of the data. Similarly, if metadata replication is in effect, the system storage pool must have two failure groups.

Disks that belong to storage pools in which write affinity is enabled can use topology vectors to identify failure domains in a shared-nothing cluster. Disks that belong to traditional storage pools must use simple integers to specify the failure group.

If this clause is specified, the value must match the failure group already in effect for the disk; `mmchnsd` cannot be used to change this value.

pool=*StoragePool*
Specifies the storage pool to which the disk is to be assigned. If this clause is specified, the value must match the storage pool already in effect for the disk; `mmchnsd` cannot be used to change this value.

device=*DiskName*
Is the block device name of the underlying disk device. This clause is ignored by the `mmchnsd` command.

thinDiskType={no | nvme | scsi | auto}
Specifies the space reclaim disk type:

no
The disk does not support space reclaim. This value is the default.

nvme
The disk TRIM capable NVMe device that supports the `mmreclaimspace` command.

scsi
The disk is a thin provisioned SCSI disk that supports the `mmreclaimspace` command.

auto
The type of the disk is either `nvme` or `scsi`. IBM Spectrum Scale will try to detect the actual disk type automatically. To avoid problems, you should replace `auto` with the correct disk type, `nvme` or `scsi`, as soon as you can.

Note: In 5.0.5, the space reclaim auto-detection is enhanced. It is encouraged to use the auto key-word after your cluster is upgraded to 5.0.5.

For more information, see the topic *IBM Spectrum Scale with data reduction storage devices* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmchnsd` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

If the disk **gpfs1nsd** is currently defined with **k145n05** as the first server and **k145n07** as the second server, and you want to replace **k145n05** with **k145n09**, create a file `./newNSDstanza` that contains:

```
%nsd: nsd=gpfs1nsd
  servers=k145n09,k148n07
```

Issue the following command:

```
# mmchnsd -F ./newNSDstanza
```

To confirm the changes, issue the following command:

```
# mmlsnsd -d gpfs1nsd
```

A sample output is as follows:

```
File system  Disk name  NSD servers
------------------------------------------------------------------------
fs2          gpfs1nsd   k145n09.ppd.pok.ibm.com,k145n07.ppd.pok.ibm.com
```

See also

- “mmchdisk command” on page 212
- “mmcrcluster command” on page 306
- “mmcrnsd command” on page 335
- “mmlsnsd command” on page 522

Location

/usr/lpp/mmfs/bin

mmchpolicy command

Establishes policy rules for a GPFS file system.

Synopsis

```
mmchpolicy Device PolicyFilename [-t DescriptiveName] [-I {yes | test}]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmchpolicy` command to establish the rules for policy-based lifecycle management of the files in a given GPFS file system. Some of the things that can be controlled with the help of policy rules are:

- File placement at creation time
- Snapshot data placement during file writes and deletes
- Replication factors
- Movement of data between storage pools
- File deletion

The `mmapplypolicy` command must be run to move data between storage pools or delete files.

Policy changes take effect immediately on all nodes that have the affected file system mounted. For nodes that do not have the file system mounted, policy changes take effect upon the next mount of the file system.

For file systems that are created at or upgraded to product version V4.1.1 or later: If there are no `SET POOL` policy rules installed to a file system by `mmchpolicy`, the system acts as if the single rule `SET POOL '`*first-data-pool*`'` is in effect, where *first-data-pool* is the firstmost non-system pool that is available for file data storage, if such a non-system pool is available. ("Firstmost" is the first according to an internal index of all pools.) However, if there are no policy rules installed and there is no non-system pool, the system acts as if `SET POOL 'system'` is in effect.

This change applies only to file systems that were created at or upgraded to V4.1.1 or later. Until a file system is upgraded, if no `SET POOL` rules are present (set by `mmchpolicy`) for the file system, all data will be stored in the `'system'` pool.

For information on GPFS policies, see the *IBM Spectrum Scale: Administration Guide*.

Parameters

Device
Specifies the device name of the file system for which policy information is to be established or changed. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**. This must be the first parameter.

PolicyFilename
Specifies the name of the file that contains the policy rules. If you specify `DEFAULT`, GPFS replaces the current policy file with a single policy rule that assigns all newly-created files to the `system` storage pool.

Options

-I {yes | test}
Specifies whether to activate the rules in the policy file *PolicyFileName*.

yes
The policy rules are validated and immediately activated. This is the default.

test
The policy rules are validated, but not installed.

-t *DescriptiveName*
Specifies an optional descriptive name to be associated with the policy rules. The string must be less than 256 characters in length. If not specified, the descriptive name defaults to the base name portion of the *PolicyFileName* parameter.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmchpolicy` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. This command validates a policy before it is installed:

```
mmchpolicy fs2 fs2.pol -I test
```

The system displays output similar to:

```
Validated policy `fs2.pol': parsed 3 Placement Rules, 0 Restore Rules,
 3 Migrate/Delete/Exclude Rules, 0 List Rules, 0 External Pool/List Rules
```

2. This command installs a policy:

```
mmchpolicy fs2 fs2.pol
```

The system displays output similar to:

```
Validated policy `fs2.pol': parsed 1 Placement Rules, 0 Restore Rules,
 0 Migrate/Delete/Exclude Rules, 1 List Rules, 1 External Pool/List Rules
Policy `fs2.pol' installed and broadcast to all nodes.
```

To confirm the change, issue this command:

```
mmlspolicy fs2
```

The system displays output similar to:

```
Policy file for file system '/dev/fs2':
  Installed by root@k155n11.kgn.ibm.com on Mon Dec 12
   16:56:31 2005.
  First line from original file 'fs2.pol' was:
/* This is the policy for the fs2 GPFS file system. */
```

See also

- “mmapplypolicy command” on page 80

- “mmlspolicy command” on page 526

Location

/usr/lpp/mmfs/bin

mmchpool command

Modifies storage pool properties.

Synopsis

```
mmchpool Device {PoolName[,PoolName...] | all}
         [--block-group-factor BlockGroupFactor]
         [--write-affinity-depth WriteAffinityDepth]
```

or

```
mmchpool Device -F PoolDescriptorFile
```

Availability

Available on all IBM Spectrum Scale editions.

When running the `mmchpool` command, the file system must be unmounted on all nodes.

Description

Use the `mmchpool` command to change storage pool properties.

Parameters

Device
Specifies the device name of the file system for which storage pool information is to be changed. File system names do not need to be fully qualified; for example, `fs0` is as acceptable as `/dev/fs0`.

***PoolName*[,*PoolName*...]**
Specifies one or more storage pools for which attributes will be changed.

all
Changes the attributes for all the storage pools in the specified file system.

--block-group-factor *BlockGroupFactor*
Specifies how many file system blocks are laid out sequentially on disk to behave like a single large block. This option only works if `--allow-write-affinity` is set for the data pool. This applies only to a new data block layout; it does not migrate previously existing data blocks.

--write-affinity-depth *WriteAffinityDepth*
Specifies the allocation policy to be used. This option only works if `--allow-write-affinity` is set for the data pool. This applies only to a new data block layout; it does not migrate previously existing data blocks.

-F *PoolDescriptorFile*
Specifies a file used to describe the storage pool attributes. The file contains one line per storage pool, in the following format:

```
%pool:name:blockSize:diskUsage:reserved:maxDiskSize:allocationType:allowWriteAffinity:writeAffinityDepth:blockGroupFactor:
```

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmchpool` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Example

For example, to change the `writeAffinityDepth` to 2 for FPO pool `pool1` of file system `fs1`, issue this command:

```
mmchpool fs1 pool1 --write-affinity-depth 2
```

To confirm the change, issue this command:

```
mmlspool fs1 pool1 -L
```

The system displays information similar to the following:

```
# mmlspool fs_1 p1 -L
Pool:
  name                   = pool1
  poolID                 = 65537
  blockSize              = 4 MB
  usage                  = dataOnly
  maxDiskSize            = 11 TB
  layoutMap              = cluster
  allowWriteAffinity     = yes
  writeAffinityDepth     = 2
  blockGroupFactor       = 128
```

See also

- “mmlspool command” on page 528

Location

`/usr/lpp/mmfs/bin`

mmchqos command

Controls Quality of Service for I/O operations (QoS) settings for a file system.

Synopsis

```
mmchqos Device --enable [--reset] [--force]
   [--fine-stats Seconds] [--pid-stats {yes|no}]
   [--stat-poll-interval Seconds] [--stat-slot-time Milliseconds]
   [[-N {Node[,Node...] | NodeFile | NodeClass}]
   [-C {all | all_remote | ClusterName[,ClusterName...]}]
   pool=PoolName[,QOSClass={nnnIOPS | unlimited}][,QOSClass=...] ...]
```

or

```
mmchqos Device --enable [--reset] [--force] -F StanzaFile

   [--fine-stats Seconds] [--pid-stats {yes|no}]
```

or

```
mmchqos Device --disable
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Attention: The `mmqos` command provides the same functionality as both the `mmchqos` command and the `mmlsqos` command and has additional features. Future QoS features will be added to the `mmqos` command rather than to either the `mmchqos` command or the `mmlsqos` command. For more information, see "mmqos command" on page 619.

With the `mmchqos` command, you can regulate I/O access to a specified storage pool by allocating shares of I/O operations to two QoS classes:

- A `maintenance` class for I/O-intensive, potentially long-running GPFS commands. Typically you assign fewer IOPS to this class to prevent the I/O-intensive commands from dominating file system performance and significantly delaying other tasks.
- An `other` class for all other processes. Typically you assign more IOPS or `unlimited` to this class so that normal processes have greater access to I/O resources and finish more quickly.

A third class, `misc`, is used to count the IOPS that some critical file system processes consume. You cannot assign IOPS to this class, but its count of IOPS is displayed in the output of the `mmlsqos` command.

When QoS is enabled, it restricts the active processes in a QoS class from collectively consuming more than the number of IOPS that you allocate to the class. It queues further I/O attempts until more I/O operations become available.

Important:

- You can allocate shares of IOPS separately for each storage pool.
- QoS divides each IOPS allocation equally among the specified nodes that have mounted the file system. See Table 19 on page 267.
- Allocations persist across unmounting and remounting the file system.
- QoS stops applying allocations when you unmount the file system and resumes when you remount it.
- When you change allocations or mount the file system, a brief delay due to reconfiguration occurs before QoS starts applying allocations.

Each QoS node collects statistics at certain intervals and periodically sends the accumulated statistics to a QoS manager node. The QoS manager node combines and processes the statistics so that they are available for reporting by the `mmlsqos` command. In larger clusters the frequency of collecting statistics and the frequency of sending accumulated statistics can generate so many messages between QoS nodes and a QoS manager node that the performance of QoS is degraded. To prevent that result, the values of two internal QoS variables are dynamically adjusted based on the number of nodes that have mounted the file system. The two internal variables are the interval between each collection of statistics and the interval between each sending of accumulated statistics. These variables apply globally to all the QoS nodes in the cluster. As the values of these two variables become greater, the frequency of statistics-related messages between QoS nodes and QoS manager nodes decreases and the impact on QoS performance diminishes.

The following table shows how QoS sets the collection interval and the message interval based on the number of nodes that have mounted the file system:

Table 18. Default intervals for collecting and sending statistics

Number of nodes that have mounted the file system	Interval between collecting statistics, in milliseconds	Interval between sending statistics to the QoS manager, in seconds
< 32	1000	5
< 64	2000	10
< 128	3000	15
< 256	4000	20
< 512	6000	30
< 1024	8000	40
< 2048	10000	50
< 4096	12000	60
< 8192	12000	60
< 16384	12000	60
16384 or more	24000	120

For example, the first row of the table shows that when the cluster contains fewer than 32 nodes, each QoS node collects statistics every 1 seconds and sends accumulated statistics to a QoS manager node every 5 seconds.

You can set the collection interval and the message interval to custom values with the `mmchqos` command line parameters **`--stat-slot-time`** and **`--stat-poll-interval`**. These custom values override the default values that are shown in the previous table and are not affected by changes in the number of nodes that have mounted the file system. For more information, see the description of these parameters later in this topic.

For more information about the `mmchqos` command, see *Setting the Quality of Service for I/O operations (QoS)* in *IBM Spectrum Scale: Administration Guide*.

Parameters

Device
The device name of the file system to which the command applies.

`--enable`
Causes QoS to start or to continue to apply IOPS allocations:

- If you are specifying the `--enable` option for the first time, then QoS sets the QoS classes of pools that you do not specify in the command to `unlimited` IOPS. For QoS to be in effect, you must enable it with IOPS allocations. For more information, see *Setting the Quality of Service for I/O operations (QoS)* in *IBM Spectrum Scale: Administration Guide*.
- Subsequent `mmchqos --enable` commands cumulatively augment the IOPS allocations.

  You can use the following commands to manage the accumulated allocations for a file system:

  – To see the accumulated allocations, issue the following command:

  ```
  # mmlsqos fs0
  ```

  – To remove the accumulated allocations, issue the following command:

  ```
  # mmchqos fs0 --reset
  ```

  – To disable the accumulated allocations, issue the following command:

  ```
  # mmchqos fs0 --disable
  ```

  – To reenable the accumulated allocations, issue the following command:

  ```
  # mmchqos fs0 --reenable
  ```

--disable
Causes QoS to stop applying IOPS allocations. Lets the file system run without any participation by QoS.

--reset
Causes QoS to set any QoS classes that you do not specify in the same command to `unlimited` IOPS.

When you enter multiple `mmchqos` commands for different storage pools, QoS typically records the settings for each pool and regulates the I/O consumption of each pool accordingly. However, with the `--reset` parameter, QoS discards the settings for all pools that are not specified in the same command and sets their QoS classes to `unlimited`. You can use this feature to reset the settings for any pools that you no longer want QoS to regulate and monitor.

--force
Causes QoS to accept an IOPS value lower than 100 IOPS.

Assigning less than 100 IOPS to a class is typically ineffective, because processes in that class run for an indefinitely long time. Therefore, the `mmchqos` command rejects IOPS values less than 100IOPS with an error message, unless you specify the --force option.

--fine-stats ***Seconds***
Specifies how many seconds of fine-grained statistics to save in memory for the `mmlsqos` command to display. The default value is 0, which indicates that no fine-grained statistics are saved. The valid range is 0 - 3840 seconds.

Note: The value that you specify for *Seconds* is mapped to a larger actual value. The `mmlsqos` command reports the actual value.

Fine-grained statistics are taken at one-second intervals and contain more information than regular statistics. For more information, see the topic "mmlsqos command" on page 530.

--pid-stats {yes|no}
Enables or disables the keeping of fine-grained statistics for each QoS program that is active on each node of the cluster. The default value is no. This parameter is effective only if `--fine-stats` is not zero.

When this parameter is disabled, statistics reflect the combined data of all the QoS processes running on each node.

[--stat-poll-interval ***Seconds*****]**
Sets the interval between each sending of accumulated statistics information by a QoS node to a QoS manager node. The value must be a multiple of the `--stat-slot-time`. For example, if the `--stat-slot-time` is 1000 milliseconds, then the `--stat-poll-interval` must be one of the following values: 1 second, 2 seconds, 3 seconds, and so on. This parameter overrides the default values that are shown in Table 18 on page 264 and is not affected by changes in the number of nodes that have mounted the file system.

To return to default operation, in which QoS dynamically adjusts the interval based on the number of files that have mounted the file system, set this attribute to 0. See Table 18 on page 264 and the description of this interval in the `Description` section of this topic.

[--stat-slot-time ***Milliseconds*****]**
Sets the interval between each collection of statistics information by a QoS node. If the value is less than 1000 milliseconds, it must be one of the following values, in milliseconds: 100, 125, 200, 250, or 500 milliseconds. This parameter overrides the values that are shown in Table 18 on page 264 and is not affected by changes in the number of nodes that have mounted the file system. See the description of this interval in the `Description` section of this topic.

To return to default operation, in which QoS dynamically adjusts the interval based on the number of files that have mounted the file system, set this attribute to 0. See Table 18 on page 264 and the description of this interval in the `Description` section of this topic.

-N {*Node***[,***Node***...] |** ***NodeFile*** **|** ***NodeClass*****}**
Specifies nodes in the local cluster to which the IOPS are to be allocated. QoS divides the allocated IOPS equally among the specified nodes that have mounted the file system.

all
All the nodes in the cluster where the command is entered.

Node**[,*****Node*****...]**
The specified nodes.

NodeFile
A file that contains a list of nodes.

NodeClass
The nodes in the specified class.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

This command does not support a *NodeClass* of `mount`.

-C {all | all_remote | ***ClusterName*****[,*****ClusterName*****...]}**
Specifies clusters to which the IOPS are to be allocated. For each cluster, the IOPS are divided equally among the nodes that have mounted the file system.

all
All clusters, including both the local cluster (the cluster where the command is entered) and all remote clusters.

all_remote
All clusters other than the one where the command was entered.

ClusterName**[,*****ClusterName*****...]**
The specified clusters.

If neither the `-N` nor the `-C` parameter is specified, QoS divides the IOPS as described in the last row of Table 19 on page 267.

Table 19. Allocation of IOPS	
Option	**Allocation of IOPS**
`-N` {`all` | *Node*[,*Node*...] | *NodeFile* | *NodeClass*}	The IOPS are divided equally among the specified nodes that have mounted the file system.
`-C` {`all` | `all_remote` | *ClusterName*[,*ClusterName*...]}	For each cluster, the IOPS are divided equally among the nodes that have mounted the file system.
Neither the `-N` nor the `-C` parameter is specified.	• For the `maintenance` class: The IOPS are divided equally among all the nodes in the local cluster that have mounted the file system. • `other` class: The IOPS are divided among all the nodes in the local cluster that have mounted the file system. Also, for each remote cluster, the IOPS are divided among all the nodes in the cluster that have mounted the file system.

pool=*PoolName*
Specifies a storage pool to whose `maintenance` or `other` class the IOPS are to be allocated. If you specify an asterisk (*) as the pool name, then the IOPS are allocated to the QoS classes of the unspecified pools. The *unspecified pools* are storage pools that you have not specified by name in any previous `mmchqos` command.

***QOSClass*={*nnn*IOPS | unlimited}**
Identifies a QoS class and allocates IOPS to it.

QOSClass
The QoS class to which IOPS are assigned. You can specify one of the following classes:

maintenance
Most I/O-intensive, potentially slow-running GPFS administration commands run in this class by default. See the list of commands that support QoS in Table 20 on page 267. Typically, you allocate fewer IOPS to this QoS class so that the commands that belong to it do not reduce overall file system performance.

When you start one of these commands, you can explicitly assign it to either QoS class. The assignment is effective only for the instance of the command that you are starting. In certain situations, you might assign one of these commands to the `other` class so that it runs faster and completes sooner.

other
All other processes that use I/O run in this class by default. Typically you assign more IOPS or `unlimited` to this class so that normal processes have greater access to I/O resources and finish more quickly.

Some I/O-intensive, potentially slow-running GPFS administration commands run in this class by default. (Currently just one: `mmchdisk`.) See the list of commands that support QoS in Table 20 on page 267. When you start one of these commands, you can explicitly assign it to either QoS class. The assignment is effective only for the instance of the command that you are starting. In certain situations, you might want to assign one of these commands to the `maintenance` class so that normal processes can finish more quickly.

The following table lists the GPFS commands that support QoS and the QoS class that the command runs in by default:

Table 20. GPFS commands that support QoS	
Commands that support QoS	**Default QoS class**
`mmadddisk`	`maintenance`
`mmapplypolicy`	`maintenance`

Table 20. GPFS commands that support QoS (continued)	
Commands that support QoS	**Default QoS class**
mmbackup	maintenance
mmchdisk	other
mmcheckquota	maintenance
mmdefragfs	maintenance
mmdeldisk	maintenance
mmdelfileset	maintenance
mmdelsnapshot	maintenance
mmdf	maintenance
mmfileid	maintenance
mmfsck	maintenance
mmimgbackup	maintenance
mmimgrestore	maintenance
mmlssnapshot	maintenance
mmrestripefs	maintenance
mmrpldisk	maintenance

***nnn*IOPS**
You can use the following values for IOPS:

- A value in the range 0IOPS - 1999999999IOPS. For an IOPS value less than 100, you must specify the `-force` option. Otherwise, QoS displays an error message like the following one:

  ```
  maintenance=99iops is not reasonable. To insist, try --force.
  ```

  QoS divides the IOPS allocation equally among the specified nodes that have mounted the file system.

  You can type nnnIOPS either with or without the trailing characters IOPS. So either of the following two examples is valid:

  ```
  (1) maintenance=400
  (2) maintenance=400IOPS
  ```

- `unlimited`: QoS does not restrict access to I/O operations.

-F *StanzaFile*
Specifies a file that contains QoS stanzas that describe allocations of IOPS. QoS stanzas have the following format:

```
%qos:
  pool=PoolName
  class=QOSClass
  iops=Value
  nodes={Node[,Node...] | NodeFile | NodeClass}
  cluster={all | all_remote | ClusterName[,ClusterName...]}
```

where:

%qos
Identifies the stanza as a QoS stanza.

pool=*PoolName*
Specifies a storage pool to whose `maintenance` or `other` class the IOPS are allocated. If you specify an asterisk (*) as the pool name, then the IOPS are allocated to the QoS classes of unspecified pools. *Unspecified pools* are storage pools that you have not specified by name in any previous `mmchqos` command.

class=*QOSClass*
Specifies a QoS class. It must be `maintenance` or `other`.

iops=*Value*
Specifies the IOPS that are to be allocated to the QoS class. QoS divides the allocated IOPS among the specified nodes that have mounted the file system.

nodes={*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies the nodes to which the IOPS are allocated.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

This command does not support a *NodeClass* of `mount`.

cluster={all | all_remote | *ClusterName*[,*ClusterName...*]}
Specifies clusters to which the IOPS are to be allocated. For each cluster, the IOPS are divided equally among the nodes that have mounted the file system.

all
All clusters, including both the local cluster (the cluster where the command is entered) and all remote clusters.

all_remote
All clusters other than the one where the command was entered.

***ClusterName*[,*ClusterName...*]**
The specified clusters.

Exit status

0
Successful completion.

Nonzero
A failure occurred.

Security

You must have root authority to run the `mmchqos` command.

The node on which you enter the command must be able to execute remote shell commands on any other administration node in the cluster. It must be able to do so without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. The following command enables QoS. If you are using the `--enable` option for the first time, then QoS sets the value of all the QoS classes to `unlimited`. If not, then the QoS classes retain their current settings:

```
# mmchqos fs0 --enable
```

2. The following command disables QoS but does not change the current allocations of IOPS:

```
# mmchqos fs0 --disable
```

3. The following command enables QoS and allocates 123 IOPS to the `maintenance` class of each unspecified pool:

```
# mmchqos fs0 --enable pool=*,maintenance=123IOPS
```

You could also type the command like this:

```
# mmchqos fs0 --enable pool=*,maintenance=123
```

4. The following command enables QoS and allocates 222 IOPS to the `maintenance` class of each unspecified pool. It also allocates 576 IOPS to the `maintenance` class of the pool `mySSDs`. You might make an allocation like this one to favor a pool of high-speed storage (`mySSDs`) that you expect to be accessed frequently. The command does not change the allocations of the QoS `other` classes:

Storage pool	maintenance class	other class
Each unspecified pool	222IOPS	Unchanged
mySSDs	567IOPS	Unchanged

```
# mmchqos fs0 --enable pool=*,maintenance=222IOPS pool=mySSDs,maintenance=567IOPS
```

5. The following command enables QoS and allocates IOPS to the classes of the unspecified pools and three named pools:

Storage pool	maintenance class	other class
Each unspecified pool	444IOPS	555IOPS
system	111IOPS	unlimited
second	222IOPS	unlimited
third	333IOPS	unlimited

The command is all on one line:

```
# mmchqos fs0 --enable pool=*,maintenance=444IOPS,other=555iops
                pool=system,maintenance=111IOPS,other=unlimited
                pool=second,maintenance=222IOPS,other=unlimited
                pool=third,other=unlimited,maintenance=333IOPS
```

6. The following command enables QoS and allocates IOPS to the classes of three named pools:

Storage pool	maintenance class	other class
Each unspecified pool	Unchanged	Unchanged
system	111IOPS	unlimited
second	222IOPS	unlimited
third	333IOPS	unlimited

The command is all on one line:

```
# mmchqos fs0 --enable pool=system,maintenance=111IOPS,other=unlimited
                pool=second,maintenance=222IOPS,other=unlimited
                pool=third,other=unlimited,maintenance=333IOPS
```

7. The following command enables QoS and allocates IOPS to both classes of the system pool. Also, because the command contains the `--reset` parameter, it sets both classes of all the other storage

pools in the file system to `unlimited`. The reset affects not only any unspecified pools, but also any named pools that are not explicitly mentioned in this command.

Storage pool	maintenance class	other class
`system`	111IOPS	`unlimited`
Each unspecified pool, if any	`unlimited`	`unlimited`
Each named pool, if any, other than `system`	`unlimited`	`unlimited`

The command is all on one line:

```
# mmchqos fs0 --enable --reset pool=system,maintenance=111IOPS,other=unlimited
```

8. The first part of the following command assigns IOPS to the QoS classes of the unspecified pools. It assigns 222 IOPS to the `maintenance` class of each unspecified pool.

   The second part of the command allocates 456 IOPS to the `other` class of the storage pool `mySAN`, rather than allocating to it the typical value `unlimited`. You might make an allocation like this one to a SAN controller that serves both GPFS and other systems.

Storage pool	maintenance class	other class
Each unspecified pool	222IOPS	`unlimited`
`mySAN`	123IOPS	456IOPS

   The command is all on one line:

```
# mmchqos fs0 --enable pool=*,maintenance=222IOPS
                pool=mySAN,other=456IOPS,maintenance=123IOPS
```

9. The following command allocates IOPS as they are specified in the stanza file `qos.stanza`:

```
# mmchqos fs0 --enable -F /tmp/qos.stanza
```

   The stanza file contains only one stanza:

```
%qos:
pool=sp1
class=maintenance
iops=800
nodes=node1,aixNodes
```

   The command divides the allocated IOPS equally among the nodes `node1` and the class `aixNodes`. Assuming that the node class contains three nodes, then 200 IOPS are assigned to each of the four nodes:

Storage pool or node	maintenance class	other class
sp1 and node1	200 IOPS	Unchanged

See also

- “mmlsqos command” on page 530
- *Setting the Quality of Service for I/O operations (QoS)* in the *IBM Spectrum Scale: Administration Guide*.

Location

`/usr/lpp/mmfs/bin`

mmclidecode command

Decodes the parseable command output field.

Synopsis

```
mmclidecode EncodedString
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The parseable output of a command might contain a colon (:) which interferes with the field delimiter character colon (:) of the command output. Therefore, the filed output that might contain colon needs to be encoded. Since, Percent-encoding, also known as URL encoding, is used to encode the output fields, the characters in the following table are also encoded if present in the output field.

Character	Encoded text
!	%21
#	%23
$	%24
&	%26
'	%27
(	%28
)	%29
*	%2A
+	%2B
,	%2C
/	%2F
:	%3A
;	%3B
=	%3D
?	%3F
@	%40
[	%5B
]	%5D

The output fields can be decoded using the **mmclidecode** command.

Note: If you run the **mmclidecode** command on non-encoded strings, the command leaves them intact.

Parameters

EncodedString
The encoded string to decode.

Exit status

0
Successful completion.

Nonzero
A failure occurred.

Security

The **mmclidecode** execution does not require root user.

Examples

1. The date is encoded due to the presence of colon (:). It can be decoded as:

```
mmclidecode "Mon Jan 30 10%3A43%3A29 2017"
Mon Jan 30 10:43:29 2017
```

2. The encoded path is decoded as follows:

```
mmclidecode "%2Fmnt%2Fgpfs0"
/mnt/gpfs0
```

Location

/usr/lpp/mmfs/bin

mmclone command

Creates and manages file clones.

Synopsis

```
mmclone snap SourceFile [CloneParentFile]
```

or

```
mmclone copy CloneParentFile TargetFile
```

or

```
mmclone split Filename [Filename...]
```

or

```
mmclone redirect Filename [Filename...]
```

or

```
mmclone show Filename [Filename...]
```

Availability

Available on all IBM Spectrum Scale editions. Available on AIX and Linux.

Description

Use the `mmclone` command to create and manage file clones. Clones are writable snapshots of individual files. Cloning a file is similar to creating a copy of a file, but the creation process is faster and more space efficient because no additional disk space is consumed until the clone or the original file is modified. The keyword specified after `mmclone` determines which action is performed:

snap
Creates a read-only snapshot of an existing file for the purpose of cloning. This read-only snapshot becomes known as the clone parent.

If only one file is specified with the `mmclone snap` command, it will convert that file to a clone parent without creating a separate clone parent file. When using this method to create a clone parent, the specified file cannot be open for writing or have hard links.

copy
Creates a file clone from a clone parent created with the `mmclone snap` command or from a file in a snapshot.

split
Splits a file clone from all clone parents.

redirect
Splits a file clone from the immediate clone parent only.

show
Displays the current status for one or more specified files. When a file is a clone, the report will show the parent inode number. When a file was cloned from a file in a snapshot, `mmclone show` displays the snapshot and fileset information.

The `Depth` field in the `mmclone show` output denotes the distance of the file from the root of the clone tree of which it is a member. The root of a clone tree has depth 0. This field is blank if the file in question is not a clone. This field is not updated when a clone's ancestor is redirected or split from

the clone tree. However, even if a clone's ancestor has been split or redirected, the depth of the clone should always be greater than that of each of its ancestors.

The maximum depth for a clone tree is 1000.

Note: The `mmclone` command does not copy extended attributes.

If a snapshot has file clones, those file clones should be deleted or split from their clone parents prior to deleting the snapshot. Use the `mmclone split` or `mmclone redirect` command to split file clones. Use a regular delete (`rm`) command to delete a file clone. If a snapshot is deleted that contains a clone parent, any attempts to read a block that refers to the missing snapshot will return an error. A policy file can be created to help determine if a snapshot has file clones.

For more information about file clones and policy files, see the *IBM Spectrum Scale: Administration Guide*.

Parameters

SourceFile
Specifies the name of a file to clone.

CloneParentFile
When *CloneParentFile* is specified with a `mmclone snap` command, it indicates the name of the read-only clone parent that will be created from *SourceFile*.

When *CloneParentFile* is specified with a `mmclone copy` command, it indicates the name of a read-only clone parent. The *CloneParentFile* can be a clone parent created with the `mmclone snap` command or a file in a snapshot.

TargetFile
Specifies the name of the writable file clone that will be created from *CloneParentFile*.

Filename
Specifies the name of one or more files to `split`, `redirect`, or `show`.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

To run the `mmclone` command, you must have read access to the source file that will be cloned, and write access to the directory where the file clone will be created.

Examples

1. To create a clone parent called `base.img` from a file called `test01.img`, issue this command:

```
mmclone snap test01.img base.img
```

To use this clone parent to create a file clone called `test02.img`, issue this command:

```
mmclone copy base.img test02.img
```

After the file clone is created, use the `mmclone show` command to show information about all `img` files in the current directory:

```
mmclone show *.img
```

The system displays output similar to the following:

```
Parent  Depth   Parent inode   File name
------  -----  --------------  ---------
```

```
    yes      0                  base.img
     no      1          148488  test01.img
     no      1          148488  test02.img
```

2. To create a file clone called `file1.clone` from a file called `file1` in the `snap1` snapshot, issue this command:

```
mmclone copy /fs1/.snapshots/snap1/file1 file1.clone
```

See also

- “mmcrsnapshot command” on page 340
- “mmdelsnapshot command” on page 383

Location

`/usr/lpp/mmfs/bin`

mmcloudgateway command

Creates and manages the cloud storage tier.

Synopsis

```
mmcloudgateway account create --cloud-nodeclass CloudNodeClass --account-name AccountName
                               --cloud-type {S3 |SWIFT3  |OPENSTACK-SWIFT | CLEVERSAFE-NEW|
AZURE}
                              {--username UserName [--pwd-file PasswordFile] |
                               --src-keystore-path SourceKeystorePath
                               --src-keystore-alias-name SourceKeystoreAliasName
                               --src-keystore-type SourceKeystoreType
                              [--src-keystore-pwd-file
SourceKeystorePasswordFile]}
                              [--tenant-id TenantID]
```

or

```
mmcloudgateway account update --cloud-nodeclass CloudNodeClass --account-name AccountName
                             [--username UserName] [--pwd-file PasswordFile]
                             [--src-keystore-path SourceKeystorePath]
                             [--src-keystore-alias-name SourceKeystoreAliasName]
                             [--src-keystore-type SourceKeystoreType]
                             [--src-keystore-pwd-file
SourceKeystorePasswordFile]
```

or

```
mmcloudgateway account delete --cloud-nodeclass CloudNodeClass --account-name AccountName
```

or

```
mmcloudgateway account list [--cloud-nodeclass CloudNodeClass]
                           [--name-list | --account-name
                           AccountName] [-Y]
```

or

```
mmcloudgateway CloudStorageAccessPoint create --cloud-nodeclass CloudNodeClass
                                               --cloud-storage-access-point-name
                                               CloudStorageAccessPointName
                                               --account-name AccountName {--URL URL |--S3| --
AZURE}
                                               [--region Region] [--mpu-part-size MPUPartsSize]
                                               [--server-cert-path ServerCertPath]
                                               [--slice-size SliceSize]
                                               [--proxy-ip ProxyIP --proxy-port
ProxyPort]
```

or

```
mmcloudgateway CloudStorageAccessPoint update --cloud-nodeclass CloudNodeClass
                                              --cloud-storage-access-point-name
                                              CloudStorageAccessPointName
                                              [--URL URL] [--region Region]
                                              [--server-cert-path ServerCertPath]
                                              [--slice-size SliceSize]
                                              [--proxy-ip ProxyIP]
                                              [--proxy-port
ProxyPort]
```

or

```
mmcloudgateway CloudStorageAccessPoint delete --cloud-nodeclass CloudNodeClass
                                              --cloud-storage-access-point-name
                                              CloudStorageAccessPointName
```

or

```
mmcloudgateway CloudStorageAccessPoint list [--cloud-nodeclassCloudNodeClass]
                                            [--name-list | --cloud-storage-access-point-name
                                            CloudStorageAccessPointName] [-Y]
```

or

```
mmcloudgateway cloudService create --cloud-nodeclass CloudNodeClass
                                   --cloud-service-name CloudServiceName
                                   --cloud-service-type {Sharing |Tiering}
                                   --account-name AccountName
```

or

```
mmcloudgateway cloudService update --cloud-nodeclass CloudNodeClass
                                   --cloud-service-name CloudServiceName [--enable |--
disable]
```

or

```
mmcloudgateway cloudService delete --cloud-nodeclass CloudNodeClass
                                    --cloud-service-name
CloudServiceName
```

or

```
mmcloudgateway cloudService list [--cloud-nodeclass CloudNodeClass]
                                 [--name-list | --cloud-service-name CloudServiceName] [--
Y]
```

or

```
mmcloudgateway keymanager create --cloud-nodeclass CloudNodeClass
                                 --key-manager-name KeyManagerName
                                 --key-manager-type {RKM | LKM }
                                 [--alias Alias]
                                 [--sklm-hostname SKLMHostname
                                  --sklm-port SKLMPort
                                  --sklm-adminuser SKLMAdminUser
                                  --sklm-groupname SKLMGroupname
                                 [--sklm-pwd-file SKLMPasswordFile]]
```

or

```
mmcloudgateway keymanager update --cloud-nodeclass CloudNodeClass
                                  --key-manager-name KeyManagerName
                                  [ --sklm-port SKLMPort]
                                  [--sklm-adminuser SKLMAdminUser]
                                  [--sklm-pwd-file SKLMPasswordFile]
                                  [--update-certificate]
```

or

```
mmcloudgateway keymanager rotate --cloud-nodeclass CloudNodeClass
                                    --key-manager-name KeyManagerName
```

or

```
mmcloudgateway keymanager list [--cloud-nodeclass CloudNodeClass]
                               [--name-list | --key-manager-name
                               KeyManagerName] [-Y]
```

or

```
mmcloudgateway containerPairSet create --cloud-nodeclass CloudNodeClass
                                      --container-pair-set-name ContainerPairSetName
                                       --cloud-service-name CloudServiceName
                                      [--scope-to-filesystem |--scope-to-fileset]
                                       --path Path
                                      [--data-container DataContainer]
                                      [--meta-container MetaContainer]
                                      [--cloud-directory-path CloudDirectoryPath]
                                      [--etag {ENABLE | DISABLE}]
                                      [--enc {ENABLE | DISABLE}]
                                      [--data-location DataLocation]
                                      [--meta-location MetaLocation]
                                      [--key-manager-name KeyManagerName
                                      [--active-key ActiveKey]]
                                      [--thumbnail-size ThumbNailSize]
                                      [--transparent-recalls {ENABLE | DISABLE}]
                                      [--destroy-event-handling {ENABLE | DISABLE}]
                                      [--policy-tmp-dir PolicyTmpDir]
                                      [--auto-spillover {ENABLE | DISABLE}]
                                      [--auto-spillover-threshold AutoSpilloverThreshold]
```

or

```
mmcloudgateway containerPairSet test --cloud-nodeclass CloudNodeClass
                                      --container-pair-set-name ContainerPairSetName
```

or

```
mmcloudgateway containerPairSet update --cloud-nodeclass CloudNodeClass
                                       --container-pair-set-name ContainerPairSetName
                                       [--etag-enable | --etag-disable]
                                       [--enc-enable | --enc-disable]
                                       [--active-key ActiveKey]
                                       [--transparent-recalls {ENABLE | DISABLE}]
                                       [--destroy-event-handling {ENABLE | DISABLE}]
                                       [--policy-tmp-dir PolicyTmpDir]
                                       [--auto-spillover {ENABLE | DISABLE}]
                                       [--auto-spillover-threshold AutoSpilloverThreshold]
```

or

```
mmcloudgateway containerPairSet delete --cloud-nodeclass CloudNodeClass
                                       --container-pair-set-name ContainerPairSetName
                                       [--policy-tmp-dir PolicyTmpDir]
```

or

```
mmcloudgateway containerPairSet list [--cloud-nodeclass CloudNodeClass]
                                     [--name-list | --container-pair-set-name
                                     ContainerPairSetName] [--Y]
```

or

```
mmcloudgateway maintenance create --cloud-nodeclass CloudNodeClass
                                  --maintenance-name MaintenanceName
                                  {--daily HH:MM-HH:MM | --weekly w:HH:MM-w:HH:MM}
```

or

```
mmcloudgateway maintenance update --cloud-nodeclass CloudNodeClass
                                  --maintenance-name MaintenanceName
                                  {--daily HH:MM-HH:MM | --weekly w:HH:MM-w:HH:MM}
```

or

```
mmcloudgateway maintenance delete --cloud-nodeclass CloudNodeClass
                                  --maintenance-name MaintenanceName
```

or

```
mmcloudgateway maintenance list --cloud-nodeclass CloudNodeClass[-Y]
```

or

```
mmcloudgateway maintenance setState --cloud-nodeclass CloudNodeClass
                                    --maintenance-name MaintenanceName
                                    --state{--disable |--enable}}
```

or

```
mmcloudgateway maintenance setFrequency --cloud-nodeclass CloudNodeClass
                                  --task {reconcile |backup |delete}
                                  --frequency {never |daily |weekly |monthly}
```

or

```
mmcloudgateway maintenance status [--cloud-nodeclass CloudNodeClass] [-Y]
```

or

```
mmcloudgateway config set --cloud-nodeclass CloudNodeClass
                          Attribute=value[,Attribute=value...]
```

or

```
mmcloudgateway config list [--cloud-nodeclass CloudNodeClass] [-Y]
```

or

```
mmcloudgateway node list [--nodeclass-sort] [-Y]
```

or

```
mmcloudgateway clientAssist {ENABLE | DISABLE}[-N {Node[,Node...]|NodeFile| NodeClass}]
```

or

```
mmcloudgateway service start [-N {alltct | Node[,Node...] | NodeFile | NodeClass}]
```

or

```
mmcloudgateway service stop [-N {alltct | Node[,Node...] | NodeFile | NodeClass}]
```

or

```
mmcloudgateway service status [-N {alltct | Node[,Node...] | NodeFile | NodeClass}]
                              [--cloud-storage-access-point-name
                               CloudStorageAccessPointName][-Y]
```

or

```
mmcloudgateway service version [-N {all | alltct | Node[,Node...] | NodeFile | NodeClass}][-Y]
```

or

```
mmcloudgateway service backupConfig --backup-file  BackupFile
```

or

```
mmcloudgateway service restoreConfig --backup-file  BackupFile
```

or

```
mmcloudgateway files migrate [-v] [--co-resident-state]
                                  [--cloud-service-name CloudServiceName] [--]
                                  File[File ...]
```

or

```
mmcloudgateway files recall [-v] [--local-cluster
                                 LocalCluster] [--owning-cluster OwningCluster] [--]
                                 File[File ...]
```

or

```
mmcloudgateway files restore [-v] [--overwrite] [--restore-stubs-only]
                                  {-F FileListFile | [--dry-run]
                                  [--restore-location RestoreLocation]
                                  [ --id ID] [--] File}
```

or

```
mmcloudgateway files delete {--delete-local-file | --recall-cloud-file | --require-local-file}
                            [--keep-last-cloud-file] [--] File[File ...]
```

or

```
mmcloudgateway files destroy [--cloud-retention-period-days CloudRetentionPeriodDays [--
preview]]
                             [--timeout Minutes] --container-pair-set-name ContainerPairSetName
                             --filesystem-path FileSystemPath
```

or

```
mmcloudgateway files reconcile --container-pair-set-name ContainerPairSetName Device
```

or

```
mmcloudgateway files cloudList {--path Path [--recursive  [--depth Depth]]
                                [--fileFile] |
                                --file-versions File | --files-usage --path Path
                                [--depth Depth] | --reconcile-status --path Path | --path Path
                                 --start YYYY-MM-DD[-HH:mm] --end YYYY-MM-DD[-HH:mm]}
```

or

```
mmcloudgateway files rebuildDB --container-pair-set-name ContainerPairSetName Device
```

or

```
mmcloudgateway files defragDB --container-pair-set-name ContainerPairSetName
```

or

```
mmcloudgateway files list File[File ...]
```

or

```
mmcloudgateway files backupDB --container-pair-set-name ContainerPairSetName
```

or

```
mmcloudgateway files checkDB --container-pair-set-name ContainerPairSetName
```

or

```
mmcloudgateway files import
                            [--cloud-sevice-name CloudServiceName]
                            [----container-pair-set-name ContainerPairSetName ]
                            [--import-only-stub]
                            [--import-metadata]
                            {[--directory Directory] |
                            [--directory-root DirectoryRoot] |
                            [--target-name TargetName]} File[File]
```

or

```
mmcloudgateway files export
                            [--cloud-sevice-name CloudServiceName]
                            [--target-name TargetName]
                            [----container-pair-set-name ContainerPairSetName]
                            [--manifest-file ManifestFile [--tag Tag]]
                            [--export-metadata [--fail-if-metadata-too-big]]
                            [--strip-filesystem-root] File[File]
```

or

```
mmcloudgateway files checkCloudObject  --container Container
                                                                  --object-name ObjectName
                                                                [--cloud-service-name
CloudServiceName][-Y]
```

or

```
mmcloudgateway files deleteCloudObject --container Container
                                                                { --object-name ObjectName| --
file File}
                                                                 [--cloud-service-name
CloudServiceName]
```

Availability

Available with IBM Spectrum Scale Advanced Edition, IBM Spectrum Scale Data Management Edition, IBM Spectrum Scale Developer Edition, or IBM Spectrum Scale Erasure Code Edition.

Description

Use the `mmcloudgateway` command to manage and administer the Transparent cloud tiering feature. This CLI has an extensive set of options organized by categories like account, service, files, etc. You can receive category-based command usage by typing the category with the help option. For example, **mmcloudgateway <category> -h** and only that category usage is displayed.

Parameters

account
Manages cloud storage accounts with one of the following actions:

create
Creates a cloud storage account.

--cloud-nodeclass CloudNodeClass
Specifies the node class that was created by using the **mmcrnodeclass** command.

--account-name AccountName
Specifies a name that uniquely identifies the cloud object storage account on the node class.

Note: No special characters are allowed in the name except for "-" and "_".

--cloud-type CloudType
Specifies the name of the object storage provider.

--username Name
Specifies the user name of the cloud object storage account. For Amazon S3 and IBM Cloud® Object Storage, it represents the access key.

Note: Skip this parameter for locked vaults.

--pwd-file PasswordFile
Specifies a file that includes the password.

Note: Skip this parameter for locked vaults.

--src-keystore-path SourceKeystorePath
Specifies the keystore path for the X.509 certificate and the private key that is used to authenticate the vault.

--src-keystore-alias-name SourceKeystoreAliasName
Specifies the alias name that needs to be imported to the Transparent cloud tiering keystore from the specified keystore.

--src-keystore-type SourceKeystoreType
Specifies the type of source keystore. Allowed keystore types are JKS, JCEKS.

--src-keystore-pwd-file SourceKeystorePasswordFile
Specifies the password file of the source keystore.

--tenant-id Tenant ID
Specifies the tenant ID for the cloud storage provider account.

Note: Optional for cloud type "S3" but mandatory for cloud types "Swift3" and "Swift-Keystone."

update
Updates the cloud storage account information.

--cloud-nodeclass CloudNodeClass
Specifies the node class that was created by using the **mmcrnodeclass** command.

--account-name AccountName
Specifies a name that uniquely identifies the cloud object storage account on the node.

Note: No special characters are allowed in the name except for "-" and "_".

--username Name
Specifies the user name of the cloud object storage account. For Amazon S3 and IBM Cloud Object Storage, it represents the access key.

Note: Skip this parameter for locked vaults.

--pwd-file PasswordFile
Specifies a file that includes the password.

Note: Skip this parameter for locked vaults.

--src-keystore-path SourceKeystorePath
Specifies the keystore path for the X.509 certificate and the private key that is used to authenticate the vault.

--src-keystore-alias-name SourceKeystoreAliasName
Specifies the alias name that needs to be imported to the Transparent cloud tiering keystore from the specified keystore.

--src-keystore-type SourceKeystoreType
Specifies the type of source keystore. Allowed keystore types are JKS, JCEKS.

--src-keystore-pwd-file SourceKeystorePasswordFile
Specifies the password file of the source keystore.

delete
Removes a cloud storage account.

--cloud-nodeclass CloudNodeClass
Specifies the node class that was created by using the **mmcrnodeclass** command.

--account-name AccountName
Specifies the unique name that was provided to the cloud object storage account on the Transparent cloud tiering node.

list
Lists the registered cloud accounts. Displays more information about the configured cloud account such as the cloud provider name, cloud provider tenant ID, cloud provider URL.

--cloud-nodeclass CloudNodeClass
Specifies the node class that was created by using the **mmcrnodeclass** command.

--name-list
When this option is specified, the output will include only the names of the cloud accounts (without other details) that are configured in a node class.

--account-name AccountName
Specifies the account name that you want to list. The output will display the details of the cloud account that is specified here.

cloudStorageAccessPoint
Manages cloud storage access points (CSAPs) with one of the following actions:

create
Creates one or more cloud storage access points.

--cloud-nodeclass CloudNodeClass
Specifies the node class that was created by using the **mmcrnodeclass** command.

--cloud-storage-access-point-name CloudStorageAccessPointName
Specifies the CSAP name that needs to be associated with the node class.

--account-name AccountName
Specifies the cloud account name that needs to be associated with the CSAP.

--url URL
Specifies the cloud endpoint with which Cloud services open up a network connection. For S3, the URL is implicit (derived from the region parameter automatically).

--region Region
Specifies the geographic region where the cloud service is installed and running.

Note: This parameter is only applicable for Swift and S3. To know the supported regions for Amazon S3, see *Amazon S3* in *IBM Spectrum Scale: Administration Guide*.

--mpu-parts-size MPUPartsSize
Specifies multi-part upload size in bytes. Any value expressed in bytes between 5 MB and 4 GB is allowed. Default being 128 MB.

--server-cert-path ServerCertPath
Specifies the certificate path for the self-signed certificates that are presented by the private object storage servers. This is required only when the cloud URL uses HTTPS.

--slice-size SliceSize
Specifies the internal unit of transferring data within Cloud service modules. Higher slice size indicates better performance. Default value is 512 KB.

--proxy-ip ProxyIP
If you require a proxy server to access your cloud over the network, specify the proxy server IP address in dotted decimal format. You must specify the `--proxy-port` option when enabling a proxy server.

Note: This supports only IPv4.

--proxy-port ProxyPort
Specifies the port number that the proxy server listens to.

update
Updates the CSAP details.

--cloud-nodeclass CloudNodeClass
Specifies the node class that was created by using the **mmcrnodeclass** command.

--cloud-storage-access-point-name CloudStorageAccessPointName
Specifies the CSAP name that needs to be associated with the node class.

--url URL
Specifies the cloud endpoint with which Cloud services open up a network connection. For S3, the URL is implicit (derived from the region parameter automatically).

--region Region
Specifies the geographic region where the cloud service is installed. Applicable only for Swift.

--server-cert-path ServerCertPath
Specifies the certificate path for the self-signed certificates that are presented by the private object storage servers. This is required only when the cloud URL uses HTTPS.

--slice-size SliceSize
Specifies the internal unit of transferring data within Cloud service modules. Higher slice size indicates better performance. Default value is 512 KB.

--proxy-ip ProxyIP
If you require a proxy server to access your cloud over the network, specify the proxy server IP address in dotted decimal format. You must specify the `--proxy-port` option when enabling a proxy server.

Note: This supports only IPv4.

--proxy-port ProxyPort
Specifies the port number that the proxy server listens to.

delete
Delete the CSAPs.

--cloud-nodeclass CloudNodeClass
Specifies the node class that was created by using the **mmcrnodeclass** command.

--cloud-storage-access-point-name CloudStorageAccessPointName
Specifies the CSAP that needs to be deleted.

list
Lists the CSAP details including account name, URL, and MPU part size.

--cloud-nodeclass CloudNodeClass
Specifies the node class that is associated with the CSAP.

--name-list
When this option is specified, the output will include only the CSAP names.

--cloud-storage-access-point-name CloudStorageAccessPointName
Specifies the CSAP whose details need to be listed.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

cloudService
Manages s a cloud service that can be used either for tiering or sharing. The cloud service caters to a specific file system or a fileset and to a specific cloud account:

create
Creates a cloud service.

--cloud-nodeclass CloudNodeClass
Specifies the node class that is associated with the cloud service.

--cloud-service-name CloudServiceName
Specifies a name to identify the cloud service.

--cloud-service-type CloudServiceType{Sharing | Tiering}
Specifies whether the cloud service is used for tiering or sharing operation. If it is used for tiering, then "Tiering" should be specified, and if it is used for sharing, then "Sharing" should be specified.

--account-name AccountName
Specifies the cloud account that is associated with the cloud service.

update
Updates the cloud service details.

--cloud-nodeclass CloudNodeClass
Specifies the node class associated with the cloud service whose details need to be updated.

--cloud-service-name CloudServiceName
Specifies the cloud service whose details need to be updated.

--enable | --disable
Specifies whether the cloud service needs to be enabled or disabled.

delete
Deletes the cloud service that no longer needs to be used.

--cloud-nodeclass CloudNodeClass
Specifies the node class associated with the cloud service that you want to delete.

--cloud-service-name CloudServiceName
Specifies the name of the cloud service that needs to be removed.

list
Lists the cloud service information such as name, cloud account name, and the type of cloud service (tiering/sharing).

--cloud-nodeclass CloudNodeClass
Specifies the node class associated with the cloud service that you want to list.

--name-list
When this option is specified, the output will include only the CSAP names.

--cloud-service-name CloudServiceName
Specifies the name of the cloud service that needs to be listed.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

keyManager
Manages a key manager for encrypting data between local file system and the cloud storage tier, with the following options:

create
Uses an SKLM key manager with Cloud services.

--cloud-nodeclass CloudNodeClass
Specifies the node class that is associated with the Cloud services.

--key-manager-name KeyManagerName
Specifies the key manager name.

--key-manager-type
Specifies the type of the key manager. If it is a remote key manager, specify "RKM," and if it is a local key manager, specify "LKM".

--alias Alias
Specifies the alias name of the local key manager.

--sklm-hostname SKLMHostname
Specifies the host name or IP address of the IBM Security Lifecycle Manager server.

--sklm-port SKLMPort
Specifies the port number on which the IBM Security Key Lifecycle Manager server listens for requests. Default value is 9080.

--sklm-adminuser SKLMAdminUser
Specifies the administrator user name of the IBM Security Key Lifecycle Manager server REST Global Admin. Default value is SKLMAdmin.

--sklm-groupname SKLMGroupname
Specifies the group user name of the IBM Security Key Lifecycle Manager server REST Global Admin.

--sklm-pwd-file SKLMPasswordFile
Specifies the password file of the IBM Security Key Lifecycle Manager server REST Global Admin.

update
Updates the key manager details.

--cloud-nodeclass CloudNodeClass
Specifies the node class that was created by using the **mmcrnodeclass** command.

--key-manager-name KeyManagerName
Specifies the key manager name.

--sklm-port SKLMPort
Specifies the port number on which the IBM Security Key Lifecycle Manager server listens for requests. Default value is 9080.

--sklm-adminuser SKLMAdminUser
Specifies the administrator user name of the IBM Security Key Lifecycle Manager server REST Global Admin. Default value is SKLMAdmin.

--sklm-pwd-file SKLMPasswordFile
Specifies the password file of the IBM Security Key Lifecycle Manager server REST Global Admin.

--update-certificate
Specifies this parameter if you want to update the REST certificate.

rotate
Rotates the existing key and creates a new SKLM key.

--cloud-nodeclass CloudNodeClass
Specifies the node class that was created by using the **mmcrnodeclass** command.

--key-manager-name KeyManagerName
Specifies the key manager name.

list
Lists the key manager information such as name, location of the certificate, admin user ID..

--cloud-nodeclass CloudNodeClass
Specifies the node class.

--name-list
When this option is specified, the output will include only the account names.

--key-manager-name KeyManagerName
Specifies the key manager name.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

clientAssist
Allows administrators to enable or disable a node with outbound cloud access as a client-assist node. This node assists in recalls from a Cloud services client node. This is a special node performing recall operations from client nodes directly, without redirecting requests to Cloud services server node.

containerPairSet
Manages the cloud containers (data and metadata) that are associated with the cloud storage accounts, with these options:

create
Creates a container pair set.

--cloud-nodeclass CloudNodeClass
Specifies the node class that is associated with the containers that are going to be created.

--container-pair-set-name ContainerPairSetName
Specifies a name that uniquely identifies the containers.

--cloud-service-name CloudServiceName
Specifies the cloud service that is going to be associated with the containers you are going to create.

--scope-to-filesystem | scope-to-fileset
Specifies whether the containers are going to be associated with a file system or a fileset. Default value is --scope-to-filesystem.

--path Path
Specifies the path to the file system or fileset.

--data-container DataContainer
Specifies a name for the data container. If you do not specify a value here, the *--container-pair-set-name* is used by default.

--meta-container MetaContainer
Specifies a name for the metadata container. If you do not specify a value here, the *--container-pair-set-name* is used by default.

--cloud-directory-path CloudDirectoryPath
Specifies the path where the database (cloud directory) is maintained.

Note: By default, the database is maintained inside the `.mcstore` folder in file system which is associated with the cloud service.

--etag-enable{ENABLE|DISABLE}
Specifies whether you want to enable an integrity check on the data that is migrated to or recalled from the cloud storage. Disabled by default.

--enc-enable{ENABLE|DISABLE}
Specifies whether you want to enable encryption on the data that is transferred to the object storage. Disabled by default.

--data-location DataLocation
Specifies the location ID of the data container. This parameter is applicable only to IBM Cloud™ Object Storage.

--meta-location MetaLocation
Specifies the location ID of the metadata container. This parameter is applicable only to IBM Cloud™ Object Storage.

--key-manager KeyManager
Specifies the key manager that needs to be chosen for encryption.

--active-key Activekey
Specifies the current active encryption key.

--thumbnail-size ThumbnailSize
Specifies the number of bytes that Transparent cloud tiering must store on the local file system for displaying thumbnail of files that are migrated to a storage tier. The value that you specify is applicable to each file in the file system that is managed by Transparent cloud tiering. Valid range is 1 - 1048576 bytes (1 MB). If you specify a value that is lower than the file system block size, then the file system block size is used. For example, if you specify a value of 128 KB and the file system block size is 256 KB, then 256 KB data of each file is stored locally and used for thumbnail.

After the thumbnail-size parameter is enabled with a file system, you can verify the setting by using the **mmcloudgateway filesystem list** command, and additionally, the thumbnails are displayed when you browse the files on Windows Explorer or a similar tool.

Note: Thumbnail is disabled by default. If you do not specify a valid value, then thumbnail is not enabled for the file system. You should judiciously make a decision to enable this option, as once enabled, you cannot disable this.

--transparent-recalls {ENABLE|DISABLE}
Specify whether or not you want to enable or disable transparent recalls for the container that you create. Enabled by default.

--destroy-event-handling {ENABLE|DISABLE}
Specify whether or not you want to enable or disable this function. It will install the policy partition to handle the destroy events when files are deleted. Enabled by default.

--policy-tmp-dir
Specify the folder where the policy file needs to be temporarily stored.

--auto-spillover {ENABLE|DISABLE}
Specify whether you want to enable or disable automatic creation of a new container, after the default limit or the specified limit in terms of number of files for an existing container is reached. Enabled by default.

--auto-spillover-threshold *AutoSpilloverThreshold*
Specify the number of files after which you want a new container to be automatically created. Default value is 100M (100,000,000).

test
Validates the cloud storage account and the CSAPs before you proceed with cloud operations.

--cloud-nodeclass CloudNodeClass
Specifies the node class that is associated with the containers that are created.

--container-pair-set-name ContainerPairSetName
Specifies a name that uniquely identifies the cloud object storage account on the node.

update
Updates the information that is used for creating a container pair set.

--cloud-nodeclass CloudNodeClass
Specifies the node class that is associated with the containers that are created.

--container-pair-set-name ContainerPairSetName
Specifies a name that uniquely identifies the cloud object storage account on the node.

--etag {ENABLE|DISABLE}
Specifies whether you want to enable an integrity check on the data that is migrated to or recalled from the cloud storage. To enable, specify "ENABLE". To disable, specify "DISABLE".

--enc {ENABLE|DISABLE}
Specifies whether you want to enable encryption on the data that is transferred to the object storage. To enable, specify "ENABLE". To disable, specify "DISABLE".

--active-key Activekey
Specifies the current active encryption key.

--transparent-recalls {ENABLE|DISABLE}
Specifies whether you want to enable or disable transparent recalls for the container pair set associated with the specified node class. To enable, specify "ENABLE". To disable, specify "DISABLE".

--destroy-event-handling {ENABLE|DISABLE}
Specifies whether you want to enable or disable the destroy event handling. To enable, specify "ENABLE". To disable, specify "DISABLE".

--policy-tmp-dir
Specify the folder where the policy file needs to be temporarily stored.

--auto-spillover {ENABLE|DISABLE}
Specify whether you want to enable or disable automatic creation of a new container, after the default limit or the specified limit in terms of number of files for an existing container is reached. Enabled by default.

--auto-spillover-threshold *AutoSpilloverThreshold*
Specify the number of files after which you want a new container to be automatically created. Default value is 100M (100,000,000).

delete
Deletes the container pair set associated with a specific node class.

--cloud-nodeclass CloudNodeClass
Specifies the node class that is associated with the containers that are created.

--container-pair-set-name ContainerPairSetName
Specifies a name that uniquely identifies the cloud object storage account on the node.

--policy-tmp-dir
Specify the folder where the policy file needs to be temporarily stored.

list
Lists the container pair set.

--cloud-nodeclass CloudNodeClass
Specifies the node class.

--cloud-service-name CloudServiceName
Specifies the name of the cloud service that needs to be listed.

--name-list
When this option is specified, the output will include only the account names.

--container-pair-set-name ContainerPairSetName
Specifies a name that uniquely identifies the cloud object storage account on the node.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

maintenance
Configure the Transparent cloud tiering for maintenance activities. For more information, see the *Setting up maintenance tasks* topic in the *IBM Spectrum Scale: Administration Guide*.

create
Creates a maintenance window, overriding the default values:

--cloud-nodeclass CloudNodeClass
Specifies the node class.

--maintenance-name MaintenanceName
Specifies a name for the maintenance window. All maintenance operations are considered within a maintenance window. For example, *daily_maintenance*.

--daily HH:MM-HH:MM
Indicates that the maintenance window is run daily. Specify the time interval in the hh:mm-hh:mm format. For example, *03:00-03:30*. If the end time equals the start time, the maintenance logic will only execute once. If the end time is less than the start time, it is assumed that the end time refers to that time on the following day. Therefore, 03:00-02:59 for instance will extend to 2:59 on the following day.

--weekly W:HH:MM-W:HH:MM
Indicates that the maintenance window is run weekly. Specify the time interval in the w:hh:mm-w:hh:mm format, where *w* represents the day of the week. Day of the week can be a number from 0 to 7 (both 0 and 7 are included and they represent Sunday). For example, *01:03:00-01:03:30*. Maintenance window 1:03:00-1:02:59 will extend until 2:59 the following week, and similarly, 1:03:00-0:03:00 will extend for 6 days (Monday through Sunday).

update
Modifies the maintenance window, overriding the previous values:

--cloud-nodeclass CloudNodeClass
Specifies the node class.

--maintenance-name MaintenanceName
Specifies a name to the maintenance window that you delete. For example, *daily_reconcile*.

--daily HH:MM-HH:MM
Indicates that the maintenance window is run daily. Specify the time interval in the hh:mm-hh:mm format. For example, *03:00-03:30*.

--weekly W:HH:MM-W:HH:MM
Indicates that the maintenance window is run weekly. Specify the time interval in the w:hh:mm- w:hh:mm format, where *w* represents the day of the week. For example, *01:03:00-01:03:30*.

delete
Deletes the maintenance window that is no longer needed.

--cloud-nodeclass CloudNodeClass
Specifies the node class.

--maintenance-name MaintenanceName
Specifies a name to the maintenance window that you create. For example, *window_for_reconcile*.

list
Lists the frequencies for the tasks and the maintenance windows for the node class.

--cloud-nodeclass CloudNodeClass
Specifies the node class associated with the maintenance window that you want to list.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

setState
Specify this option to change the state of a maintenance window that is created for a specified container. For example, you can have multiple maintenance tasks for a container but can disable or enable them according to your requirements. If a maintenance task is disabled, this task will no more be run from the next maintenance window. If a maintenance window is disabled, maintenance operations will not be executed for this window. Once enabled, maintenance will begin executing at the start of the window.

--cloud-nodeclass CloudNodeClass
Specifies the node class associated with the maintenance window whose state needs to be changed.

--maintenance-name MaintenanceName
Specifies the name of the maintenance window.

--state {disable | enable}
Specify one of the options listed here (disable or enable).

setFrequency
Use this option to modify the default frequency of the maintenance operations such as reconcile, backup, and delete. By default, reconcile is done monthly, backup is done weekly, and deletion is done daily. Use this option to change the frequency of any of these operations. For example, you want to run the reconcile and delete operations as per the default schedule but want to change the backup frequency to monthly. In this case, you can set the frequency of the backup operation to "monthly".

--cloud-nodeclass CloudNodeClass
Specifies the node class associated with the maintenance task.

--task {reconcile | backup | delete}
Specifies the name of the maintenance task whose frequency needs to be changed.

--frequency {never|daily|weekly|monthly}
Specify one of the options listed here.

status
Lists the summary of the maintenance schedule for a specified node class.

--cloud-nodeclass CloudNodeClass
Specifies the node class for which to display the maintenance status information.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

config
Configures and tunes the Transparent cloud tiering node parameters with one of the following actions:

set
Sets the following system parameters, overriding the default values:

--cloud-nodeclass CloudNodeClass
Specifies the node class.

Attribute=value
Specifies the attribute that you want to change and the value that you want to set. For example, if you want to change the default value of the recalls-thread attribute and set it to 20, specify "recalls-threads=20." Similarly, you can set the value of other attributes also.

Note: If you want to set an attribute back to its default value, specify "DEFAULT" as the value of the attribute. For example, if you want to set the value of recalls-thread attribute back to its default value, specify, "recalls-thread=DEFAULT".

For a list of available attributes and their description, see the *Tuning Cloud services parameters* topic in the *IBM Spectrum Scale: Administration Guide*.

list
Lists the current configurations such as IP address, port number, thread-pool size, tracing level, slice size.

--cloud-nodeclass CloudNodeClass
Specifies the node class that was created by using the **mmcrnodeclass** command.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

node
Enables administrators to manage registration of Cloud services within a cluster and also display the node class the nodes are part of.

list
Lists the nodes that are identified for enabled for Cloud services.

--nodeclass-sort
Use this option to display the node list sorted according to the node class.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

service
Manages the Transparent cloud tiering service with these options:

start
Starts the Transparent cloud tiering service for a node or set of nodes and make the service available for file movement.

-N
Specifies the nodes.

alltct
Indicates that the service will be stopped on all Transparent cloud tiering nodes within the cluster.

Node[,Node...]
Specifies the list of nodes where the service needs to be started.

NodeFile
Specifies a file, containing the list of nodes where the service needs to be started.

NodeClass
Specifies the node class.

stop
Stops the Transparent cloud tiering service for a node or set of nodes.

-N
Specifies the nodes.

alltct
Indicates that the service will be stopped on all Transparent cloud tiering nodes within the cluster.

Node[,Node...]
Specifies the list of nodes where the service needs to be stopped.

NodeFile
Specifies a file, containing the list of nodes where the service needs to be started.

NodeClass
Specifies the node class.

status
Displays detailed status of the Transparent cloud tiering service including running state of the daemon service, cloud account name, and its connectivity status. For more information on various statuses that are associated with the Transparent cloud tiering service, see the "Transparent cloud tiering service status description" topic in the *IBM Spectrum Scale: Problem Determination Guide*.

-N
Specifies the nodes.

alltct
Indicates that the service will be stopped on all Transparent cloud tiering nodes within the cluster.

Node[,Node...]
Specifies the list of nodes where the status of the service needs to be checked.

NodeFile
Specifies a file, containing the list of nodes where the status of the service needs to be started.

NodeClass
Specifies the node class.

--cloud-storage-access-point-name
Specifies the CSAP. The report will include the status of all Cloud services associated with the specified CSAP.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

version
Displays the Transparent cloud tiering version number associated with each node in a node class. This also includes the type of the node (server or client).

Note: This command can run on all nodes in the cluster and will display if a node is a cloud node or not. If it is, it will show more column-based details. This makes this command different from others since normally the commands act only on cloud nodes for report generation.

-N
Use this option before specifying any node or node class.

all
Indicates that the service version will be displayed for all nodes.

alltct
Indicates that the service version will be displayed for all Transparent cloud tiering nodes within the cluster.

Node[,Node...]
Specifies the list of nodes for which the versions need to be checked.

NodeFile
Specifies a file, containing the list of nodes for which the versions need to be started.

NodeClass
Specifies the node class whose version is displayed at a cluster level.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

backupConfig
Backs up the Cloud services configuration from the Clustered Configuration Repository (CCR). For more information, see the *Backing up the Cloud services configuration* topic in the *IBM Spectrum Scale: Administration Guide*.

--backup-file BackupFile
Specifies a name to be used for the backup file. The system generates a tar file by pulling all Cloud services-specific configuration files from the Clustered Configuration Repository (CCR). The backed up file will be stored on your local machine at a specified location. The backupConfig command also accepts a location on a GPFS file system that would be available on all GPFS nodes.

restoreConfig
Restores the Cloud services configuration data in the event of any system outage or crash. For more information, see the *Restoring the Cloud services configuration* topic in the *IBM Spectrum Scale: Administration Guide*.

--backup-file BackupFile
Specifies the file name of the backed-up file including the path. The system restores the Cloud services-specific configuration setting to the CCR by using this file.

files
Manages files, with the following options:

migrate
Migrates the specified files to the cloud storage tier.

-v
Specifies the verbose message.

--co-resident-state
Indicates that the files are migrated in the co-resident status, which means that files will be available both locally and on the cloud after migration. You can open such files from the local file system without recalling them from the cloud storage tier.

--File [File...]
Specifies multiple files that need to be migrated to the cloud storage tier. This parameter must be a complete file name. It cannot be a fragment of a file name and it cannot be a path.

recall
Recalls the specified files from the cloud storage tier.

-v
Specifies the verbose message.

--local-cluster
Indicates that cluster from where the recall is done.

--remote-cluster
Indicates the remote cluster.

--File [File...]
Specifies multiple files that need to be recalled from the storage tier. This parameter must be a complete file name. It cannot be a fragment of a file name and it cannot be a path.

restore
Restores a file or list of files from the cloud storage tier when the local files are lost. The files to be restored along with their options can be either specified at the command line, or in a separate file provided by the *-F* option. For more information, see the *Restoring files* topic in the *IBM Spectrum Scale: Administration Guide*.

-v

Specifies the verbose message.

--overwrite
Overwrite the files if needed. If this option is not set, files will not be overwritten, and the files that are retrieved from the cloud will remain in temporary locations.

--restore-stubs-only
Restores only the file stubs.

-F
Loads file arguments from the given file name.

--dry-run
Queries the local database and prints what would have been sent to the server. Does not contact the server. This is intended for debugging.

--restore-location RestoreLocation
Specifies the target location of the files to be restored.

--id Id
Specifies the version ID of a file if the file has multiple versions.

File
Specifies the files to be restored.

delete
Deletes the specified files or file sets.

--delete-local-file
Deletes the local files and the corresponding cloud objects.

--recall-cloud-file
Recalls the files from the cloud before they are deleted on the cloud. The status of local files becomes resident after the operation.

--require-local-file
Removes the extended attributes from a co-resident file and makes it resident, without deleting the corresponding cloud objects. The option requires the file data to be present on the file system and will not work on a non-resident file.

--keep-last-cloud-file
This option deletes all the versions of the file except the last one from the cloud. For example, if a file has three versions on the cloud, then versions 1 and 2 are deleted and version 3 is retained.

--File [File...]
Specifies multiple files. This parameter must be a complete file name. It cannot be a fragment of a file name and it cannot be a path.

destroy
Manually cleans up the cloud objects of the deleted files before the retention period expires. This cleanup will occur for all objects from the root of the file system provided by the `--filesystem-path` option. For more information, see the *Deleting cloud objects* topic in the *IBM Spectrum Scale: Administration Guide*.

--cloud-retention-period-days
Specifies the number of days for which the deleted files from the file system need to be retained on the cloud. For example, you delete 100 files from the file system and need to keep them on the cloud for 20 days, specify the value as 20.

--preview
Displays how many objects will be cleaned up and how much space will be reclaimed.

--timeout
Specifies the duration (in minutes) for which the command should run. If this value is not specified, the command will run until all the candidate files are deleted, and it can be resource-intensive.

--container-pair-set-name
Specifies the cloud container where the objects are stored.

--filesystem-path
The path to the file system where the files are migrated from. All objects under the root of the file system of the specified file system path will be cleaned.

reconcile
Reconciles files between your file system and the cloud storage tier. For more information, see the *Reconciling files between IBM Spectrum Scale file system and the cloud storage tier* topic in the *IBM Spectrum Scale: Administration Guide*.

--container-pair-set-name
Specifies the cloud container where the objects are stored.

Device
Specifies the device name associated with the file system.

cloudList
Lists the files on the cloud.

--path Path
Lists files and directories under the specified path.

--recursive
List all files in all directories under the current directory.

--depth Depth
List directories up to the specified depth under the specified path. Default is to list up to the full depth. Specify 0 to list only the current directory.

--file [File]
Specifies the names of the files that need to be listed. This parameter must be a complete file name. It cannot be a fragment of a file name and it cannot be a path.

--file-versions File
Displays information about all versions of the files specified by the full path.

--files-usage --path Path
Displays cloud data and metadata space usage under the specified path.

--reconcile-status --path Path
Displays the progress of the reconcile operation.

--start YYYY-MM-DD[-HH:mm]
Specifies the starting time.

--end YYYY-MM-DD[-HH:mm]
Specifies the ending time.

backupDB
Backs up the Transparent cloud tiering database to the cloud storage tier. For more information, see the *Backing up the Cloud services database to the cloud* topic in the *IBM Spectrum Scale: Administration Guide*.

--container-pair-set-name
Specifies the container associated with the database that needs to be backed up.

checkDB
Verifies the integrity of the Transparent cloud tiering database after a power outage or a system crash. For more information, see the *Checking the Cloud services database integrity* topic in the *IBM Spectrum Scale: Administration Guide*.

--container-pair-set-name
Specifies the container associated with the database that needs to be verified.

rebuildDB
Manually rebuilds the database.

--container-pair-set-name
Specifies the container associated with the database that needs to be rebuilt.

Device
Specifies the device name associated with the file system whose database is corrupted and which is in need of manual recovery.

defragDB
Defragment the database and release the capacity occupied by the empty spaces.

--container-pair-set-name
Specifies the container associated with the database that needs to be rebuilt.

list
Lists the files and the associated states.

--File [File]
Specifies the names of the files that need to be listed. This parameter must be a complete file name. It cannot be a fragment of a file name and it cannot be a path.

import
Imports data from a storage server.

--cloud-service-name
Specifies the cloud service.

--container Container
Specifies the name of the cloud container to import from. If no container option is specified, the default configured container name is used.

--import-only-stub
Creates only a stub file, data in the file is not imported.

--import-metadata
Attempts to restore metadata of the file from the cloud object. The data is in IBM Spectrum Scale format. The cloud object must have been exported using the `--export-metadata` option. If metadata is not attached to the cloud object, this option has no effect.

--directory
Imports files into the given directory using only the file name from the cloud. Mutually exclusive with the `--target-name` and `--directory-root` options.

--directory-root
Imports files starting at the given directory, keeping the cloud naming hierarchy intact. Mutually exclusive with the `--directory` and `--target-name` options.

--target-name
Imports a single file from the cloud to the specified target name. Mutually exclusive with the `--directory` and `--directory-root` options.

--File [File]
Specifies the names of the files that need to be imported. This parameter must be a complete file name. It cannot be a fragment of a file name and it cannot be a path.

export
Exports files to the cloud.

--tag Tag
Specifies an optional identifier to associate with the files. This ID will be stored in the manifest file if one is specified.

--target-name TargetName
Export a single file to the cloud to the specified target name.

--container Container
Specifies the name of the cloud container to export to. If no container option is specified, the default configured container name will be used.

--manifest-file ManifestFile
Specifies a manifest file that will contain an entry for each file exported to the cloud. Entries will be in the CSV format of : Tag, Container, TimeStamp of Blob on cloud, or file name.

--export-metadata
Attempts to attach a file's metadata to the cloud object. This is IBM Spectrum Scale specific data and format, and it contains the user-defined attributes, ACLs, etc. A file exported with this option can be fully restored by the corresponding import command. This metadata is stored in the blob metadata, and as such there is limited space available, and the metadata might not be written if it is too large.

--fail-if-metadata-too-big
If the metadata of the file is very large, it causes the entire export to fail. Valid only with the `--export-metadata` option.

--strip-filesystem-root
Removes the root of the IBM Spectrum Scale file system from the name as stored on the cloud. This could be used to export `/filesystem1/dir/file` and then import that file into a differently named file system root directory.

--File [File]
Specifies the names of the files that need to be exported. This parameter must be a complete file name. It cannot be a fragment of a file name and it cannot be a path.

checkCloudObject
Displays true if the specified object or container is present on cloud, false otherwise. If you specify the cloud service, the container associated with the cloud service is used to search the object. Use the -Y option for parsable output.

deleteCloudObject
Deletes the specified object or container on cloud. If you specify the cloud service, the container associated with the cloud service is used to search the object for deletion.

Security

You must have root authority to run the `mmcloudgateway` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To view the registered nodes in a node class, issue the following command:

```
# mmcloudgateway node list
```

A sample output is as follows:

```
Node  Cloud node name                 Cloud Node Class
-----------------------------------------------------
    8  c35f1m4n09.gpfs.net             CloudNodesClass1
    9  c34f2n06.gpfs.net               CloudNodesClass2
```

2. To start the Transparent cloud tiering service on the node class *TCTNodeClass1*, issue the following command:

```
# mmcloudgateway service start -N TCTNodeClass1
```

A sample output is as follows:

```
mmcloudgateway: Sending the command to node c350f1u1b4.
Starting the Transparent Cloud Tiering service...
mmcloudgateway: The command completed on node c350f1u1b4.

mmcloudgateway: Command completed.
```

3. To verify the status of the Transparent cloud tiering service, issue the following command:

```
# mmcloudgateway service status --cloud-storage-access-point-name swift-point
```

A sample output is as follows:

```
Cloud Node Class:  tct
======================

 Cloud Service                Status      Reason
-----------------------------------------------
 swift-service                ENABLED

                             Server     Account /   Container /
 Node  Daemon node name      Status     CSAP        File System/Set   Status    Reasons
-------------------------------------------------------------------------------------------
    1  vm.pk.sglabs.ibm.com  STARTED    swift-account  swift-pair
                                        swift-point   /gpfs/         ONLINE
```

4. To create a cloud storage account with IBM Cloud Object Storage version 3.7.2 and above as cloud type, issue the following command:

```
# mmcloudgateway account create --cloud-nodeclass TCTNodeClass1 --account-name tctenew
--cloud-type cleversafe-new --username "XYZ" --pwd-file PFile
--cloud-url http://192.0.2.0
```

A sample output is as follows:

```
Configured account options from TCTNodeClass1:
----------------------------------------------
Cloud Provider Name: tctnew
Cloud Provider Tenant Id: admin
Cloud Provider URL: http://192.0.2.0:5000/v2.0
Cloud Provider Type: swift-keystone
Cloud Provider User Name: admin
Cloud Provider Enabled: true
Filesystem: ga
Container: gajan
Metadata Container: gajan.meta
```

5. To create a cloud account for the S3 cloud type, issue a command similar to this:

```
# mmcloudgateway account create --cloud-nodeclass TCTNodeClass1 --account-name cloudtest
 --cloud-type s3 --username admin --pwd-file MyFile
```

A sample output is as follows:

```
mmcloudgateway: Sending the Transparent Cloud Tiering request to
the first successful server.
```

```
mmcloudgateway: This may take a while...
mmcloudgateway: Command completed successfully on c350f2u18.
mmcloudgateway: Command completed.
```

6. To create a cloud account for deploying a WORM solution by using locked vaults, issue a command like the following:

```
# mmcloudgateway account create --cloud-nodeclass NodeClass1 --cloud-name cloudtest
--cloud-type CLEVERSAFE-NEW --cloud-url https://198.51.100.0 --enable true
--server-cert-path /root/cleversafe.pk.stg.ibm.com --src-keystore-path
/root/test/testalias.ssl/testalias.jks --src-keystore-alias-name testalias
--src-keystore-type JKS --src-keystore-pwd-file /root/pwd/file.txt
```

A sample output is as follows:

```
mmcloudgateway: Sending the command to the first successful node
starting with c350f3u30
mmcloudgateway: This may take a while...

Note: Please ensure to keep a backup of the Source Key Store used to import
the private key and certificates.
Transparent Cloud Tiering will remove the private key and certificate from
the trust store if the account delete command is run.
mmcloudgateway: Command completed successfully on c350f3u30.
mmcloudgateway: You can now delete the password file '/root/pwd/file.txt'
mmcloudgateway: Command completed.
```

7. To update a cloud account for a locked vault, issue the following command:

```
# mmcloudgateway account update  --cloud-nodeclass NodeClass1 --cloud-name cloudtest
--src-keystore-path /root/test/testalias.ssl/testalias.jks  --src-keystore-alias-name
testaliasnew --src-keystore-type JKS --src-keystore-pwd-file /root/pwd/file.txt
```

A sample output is as follows:

```
mmcloudgateway: Sending the command to the first successful node starting with
vm641.pk.slabs.ibm.com
mmcloudgateway: This may take a while...

Note: Please ensure to keep a backup of the Source Key Store used to import the
private key and certificates.
Transparent Cloud Tiering will remove the private key and certificate from the
trust store if the account delete command is run.
mmcloudgateway: Command completed successfully on jupiter.pk.slabs.ibm.com.
mmcloudgateway: You can now delete the password file '/root/pwd'
mmcloudgateway: Command completed.
```

8. To list the cloud accounts for all node classes, issue the following command:

```
# mmcloudgateway account list --cloud-nodeclass tct
```

A sample output is as follows:

```
Configured account options from node class tct:
-----------------------------------------------
  accountName                   :  swift
  cloudType                     :  openstack-swift
  userName                      :  admin
  tenantId                      :  admin
```

9. To list all the cloud accounts configured for the node class, *cloud1*, issue the following command:

```
# mmcloudgateway account list --cloud-nodeclass cloud1
```

A sample output is as follows:

```
Configured 'accountName' options from node class cloud1:
-------------------------------------------------------
  accountName                   :  new
  cloudType                     :  openstack-swift
```

```
  userName                      :  admin
  tenantId                      :  admin
```

10. To list only the names of the cloud accounts configured for all node classes present in the cluster, issue the following command:

```
# mmcloudgateway account list --name-list

Configured 'accountName' names from node class cloud1:
------------------------------------------------------
  Swift_cloud1

Configured 'accountName' names from node class cloud2:
------------------------------------------------------
  S3_cloud2
```

11. To list the current cloud configuration, issue the following command:

```
# mmcloudgateway config list NodeClass1
```

A sample output is as follows:

```
aonfigured config options from node class NodeClass1:
-----------------------------------------------
  cloud-retention-period-days     :  30
```

12. To migrate a file (file1) to the configured cloud storage tier, issue the following command:

```
# mmcloudgateway files migrate file1
```

A sample output is as follows:

```
mmcloudgateway: Command completed.
```

13. To migrate multiple files (file1 and file2) to the configured cloud storage tier, issue the following command:

```
# mmcloudgateway files migrate file1 file2
```

A sample output is as follows:

```
mmcloudgateway: Command completed.
```

14. To verify that the file is migrated to the configured cloud storage tier, issue the following command:

```
# mmcloudgateway files list file1
```

A sample output is as follows:

```
File name    : /gpfs/girish/file1
On-line size : 45
Used blocks  : 0
Data Version : 1
Meta Version : 1
State        : Non-resident
Base Name    : 7448805A60ED1970.17F2AFD45704E1E4.52E20457CA532F09.
0000000000000000.57B76BDD.000000000000100B
```

Note: The **State** is displaying as **Non-resident**. This means that the file is successfully migrated to the cloud storage tier.

15. To recall a file from the configured cloud storage tier, issue the following command:

```
# mmcloudgateway files recall file1
```

A sample output is as follows:

```
mmcloudgateway: Command completed.
```

Note: If you run the **mmcloudgateway filesystem list** file1 command, the value of the **State** attribute is displayed as **Co-resident**. This means that the file is successfully recalled.

16. To recall multiple files (file1 and file2) from the configured cloud storage tier, issue the following command:

```
# mmcloudgateway files recall file1 file2
```

A sample output is as follows:

```
mmcloudgateway: Command completed.
```

17. To delete a cloud storage account, issue the following command:

```
# mmcloudgateway account delete --cloud-nodeclass TCTNodeClass1 --account-name mycloud
```

A sample output is as follows:

```
mmcloudgateway: Sending the Transparent Cloud Tiering request to the first successful
server.
mmcloudgateway: This may take a while...
mmcloudgateway: Command completed successfully on c350f2u18.
mmcloudgateway: Command completed.
```

18. To back up the Cloud services database associated with the container, *cpair1*, issue the following command:

```
# mmcloudgateway files backupDB --container-pair-set-name cpair1
```

A sample output is as follows:

```
mmcloudgateway: Command completed
```

19. To verify the integrity of the Cloud services database associated with the container, *cpair1*, after a system crash or an outage, issue the following command:

```
# mmcloudgateway files checkDB --container-pair-set-name cpair1
```

A sample output is as follows:

```
CheckDB returned OK.
mmcloudgateway: Command completed.
```

20. To back up the Transparent cloud tiering configuration data to a file called *tctbackup*, issue the following command:

```
# mmcloudgateway service backupConfig --backup-config-file tctbackup
```

A sample output is as follows:

```
Backup Config File List:
    [mmcloudgateway.conf  -  Retrieved]
    [_tctkeystore.jceks   -  Retrieved]
    [_cloud.settings  -  Retrieved]
    [_cloud2.settings  -  Retrieved]

mmcloudgateway: Creating the backup tar file...
mmcloudgateway: Backup tar file complete.  The file is '/tmp/tctbackup.31187.tar'.
mmcloudgateway: The backup file should be archived in a safe location.
mmcloudgateway: Command completed.
```

21. To restore the configuration to the CCR file by using the backed-up file, *tctbackup.31187.tar*, issue the following command:

```
# mmcloudgateway service restoreConfig /tmp/tctbackup.31187.tar
```

A sample output is as follows:

```
You are about to restore the TCT Configuration settings to the CCR.
Any new settings since the backup was made will be lost.
The TCT servers should be stopped prior to this operation.

Do you want to continue and restore the TCT cluster configuration?
Enter "yes" to continue:
```

22. To export a local file named `/dir1/dir2/file1` to the cloud and store it in a container named `MyContainer`, issue the following command:

```
# mmcloudgateway files export --ContainerPairSetName MyContainer
--tag MRI_Images --export-metadata
--manifest-file /dir/ManifestFile /dir1/dir2/file1
```

Note: A manifest file will be created, and the object exported to the cloud will have an entry in that manifest file, tagged with `MRI_Images`.

23. To import files from the cloud, issue the following command(s):

```
# mmcloudgateway files import --directory /localdir /dir1/dir2/file1
```

```
# mmcloudgateway files import --directory-root /localdir /dir1/dir2/file1
```

Note: This command creates a local directory structure as necessary when importing the file from the cloud. If the `--directory` option is specified, only the file name of the cloud object is used.

24. To check the Transparent cloud tiering service version of the node class, `TCTNodeClass1`, issue the following command:

```
# mmcloudgateway service version -N TCTNodeClass1
```

A sample output is as follows:

```
Cluster minReleaseLevel:   5.0.1.0
Node  Daemon node name       TCT Type          TCT Version     Equivalent Product Version
-------------------------------------------------------------------------------------------
    3  vmip53.gpfs.net       Server            1.1.5
5.0.1
    2  c40bbc1xn13.gpfs.net  Non TCT Node      --              --
    5  vmip51.gpfs.net       Server            1.1.5           5.0.1
```

25. To reconcile files between the file system, `fs1`, and the container pair set, `contain1`, issue the following command:

```
# mmcloudgateway files reconcile --container-pair-set-name contain1 fs1
```

A sample output is as follows:

```
Processing /fs1
Wed Jun 28 10:34:28 EDT 2017  Reconcile started.
Wed Jun 28 10:34:28 EDT 2017  Creating snapshot of the File System...
Wed Jun 28 10:34:28 EDT 2017  Running policy on Snapshot
to generate list of files to process.
Wed Jun 28 10:36:23 EDT 2017  Removing snapshot.
Wed Jun 28 10:36:27 EDT 2017  Reconcile is using a deletion retention
period of 30 days.
Wed Jun 28 10:36:27 EDT 2017  Reconcile will be processing 5043 inode entries.
Wed Jun 28 10:36:27 EDT 2017  Processed 463 entries out of 5043.
Wed Jun 28 10:36:27 EDT 2017  Processed 921 entries out of 5043.
Wed Jun 28 10:36:27 EDT 2017  Processed 1372 entries out of 5043.
Wed Jun 28 10:36:27 EDT 2017  Processed 1824 entries out of 5043.
```

```
Wed Jun 28 10:36:27 EDT 2017  Processed 2264 entries out of 5043.
Wed Jun 28 10:36:27 EDT 2017  Processed 2726 entries out of 5043.
Wed Jun 28 10:36:27 EDT 2017  Processed 3161 entries out of 5043.
Wed Jun 28 10:36:27 EDT 2017  Processed 3603 entries out of 5043.
Wed Jun 28 10:36:27 EDT 2017  Processed 4032 entries out of 5043.
Wed Jun 28 10:36:27 EDT 2017  Processed 4471 entries out of 5043.
Wed Jun 28 10:36:27 EDT 2017  Processed 4912 entries out of 5043.
Wed Jun 28 10:36:27 EDT 2017  Processed 4953 entries out of 5043.
Wed Jun 28 10:36:27 EDT 2017  Processed 5004 entries out of 5043.
Wed Jun 28 10:36:27 EDT 2017  Processed 5043 entries out of 5043.
Wed Jun 28 10:36:28 EDT 2017  Reconcile found 1 files that had
been migrated and were not in the directory.
Wed Jun 28 10:36:28 EDT 2017  Reconcile detected 0 deleted files
that were deleted more than 30 days ago.
Wed Jun 28 10:36:28 EDT 2017  Reconcile detected 5043 migrated files
that have been deleted from the local
file system, but have not been deleted from object storage because
they are waiting for their retention policy
time to expire.
Wed Jun 28 10:36:28 EDT 2017  Please use the 'mmcloudgateway files cloudList'
command to view the progress of the deletion of the cloud objects.
Wed Jun 28 10:36:29 EDT 2017  Reconcile successfully finished.
mmcloudgateway: Command completed.
```

See also

- “mmchconfig command” on page 170
- “mmlscluster command” on page 492
- “mmchnode command” on page 244
- “mmlsconfig command” on page 495
- “mmnfs command” on page 560
- “mmobj command” on page 573
- “mmsmb command” on page 709
- “mmuserauth command” on page 738

Location

/usr/lpp/mmfs/bin

mmcrcluster command

Creates a GPFS cluster from a set of nodes.

Synopsis

```
mmcrcluster -N {NodeDesc[,NodeDesc...] | NodeFile}
   [ [-r RemoteShellCommand] [-R RemoteFileCopyCommand] |
   --use-sudo-wrapper [--sudo-user UserName] ]
   [-C ClusterName] [-U DomainName] [-A]
   [-c ConfigFile | --profile ProfileName]
```

Note: The primary and secondary configuration server functionality is deprecated and will be removed in a future release. The default configuration service is CCR. The following parameters have been removed from the `mmcrcluster` command: `--ccr-enable`, `--ccr-disable`, `-p PrimaryServer`, and `-s SecondaryServer`. For more information, see the "mmchcluster command" on page 165.

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmcrcluster` command to create a GPFS cluster.

Upon successful completion of the `mmcrcluster` command, the **/var/mmfs/gen/mmsdrfs** and the **/var/mmfs/gen/mmfsNodeData** files are created on each of the nodes in the cluster. Do not delete these files under any circumstances. For more information, see *Quorum* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Follow these rules when creating your GPFS cluster:

- While a node may mount file systems from multiple clusters, the node itself may only be added to a single cluster using the `mmcrcluster` or `mmaddnode` command.
- The nodes must be available for the command to be successful. If any of the nodes listed are not available when the command is issued, a message listing those nodes is displayed. You must correct the problem on each node and issue the `mmaddnode` command to add those nodes.
- Designate at least one but not more than eight nodes as quorum nodes. How many quorum nodes altogether you will have depends on whether you intend to use the node quorum with tiebreaker algorithm or the regular node based quorum algorithm. For more information, see *Quorum* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.
- After the nodes are added to the cluster, use the `mmchlicense` command to designate appropriate GPFS licenses to the new nodes.
- Clusters that will include both UNIX and Windows nodes must use **ssh** and **scp** for the remote shell and copy commands. For more information, see *Installing and configuring OpenSSH on Windows nodes* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.
- Carefully consider the remote execution and remote copy tooling you want to use within your cluster. Once a cluster has been created, it is complicated to change, especially if additional nodes are added. The default tools as specified under `-r` *RemoteShellCommand* and `-R` *RemoteFileCopyCommand* by default use `/usr/bin/ssh` and `/usr/bin/scp` respectively. For more information, see *GPFS cluster creation considerations* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.
- A cluster is created with the **sdrNotifyAuthEnabled** parameter set to **yes**. This parameter specifies whether to authenticate the notify RPCs related to deadlock detection and amelioration, node overload, and node expel. System administrators can use the **mmchconfig** command to change the value of the **sdrNotifyAuthEnabled** parameter to **no**. For more details about the **sdrNotifyAuthEnabled** parameter, see *mmchconfig command* in *IBM Spectrum Scale: Command and Programming Reference*.

Parameters

-N *NodeDesc*[,*NodeDesc*...] | *NodeFile*
Specifies node descriptors, which provide information about nodes to be added to the cluster.

NodeFile
Specifies a file containing a list of node descriptors, one per line, to be added to the cluster.

***NodeDesc*[,*NodeDesc*...]**
Specifies the list of nodes and node designations to be added to the GPFS cluster. Node descriptors are defined as:

```
NodeName:NodeDesignations:AdminNodeName
```

where:

NodeName
Specifies the host name or IP address of the node for GPFS daemon-to-daemon communication. For hosts with multiple adapters, see the *IBM Spectrum Scale: Administration Guide* and search on *Using remote access with public and private IP addresses*.

The host name or IP address must refer to the communication adapter over which the GPFS daemons communicate. Aliased interfaces are not allowed. Use the original address or a name that is resolved by the **host** command to that original address. You can specify a node using any of these forms:

- Short host name (for example, h135n01)
- Long, fully-qualified, host name (for example, h135n01.ibm.com)
- IP address (for example, 7.111.12.102). IPv6 addresses must be enclosed in brackets (for example, [2001:192::192:168:115:124]).

Regardless of which form you use, GPFS will resolve the input to a host name and an IP address and will store these in its configuration files. It is expected that those values will not change while the node belongs to the cluster.

NodeDesignations
An optional, "-" separated list of node roles:

- **manager** | **client** – Indicates whether a node is part of the node pool from which file system managers and token managers can be selected. The default is **client**.
- **quorum** | **nonquorum** – Indicates whether a node is counted as a quorum node. The default is **nonquorum**.

AdminNodeName
Specifies an optional field that consists of a node name to be used by the administration commands to communicate between nodes. If *AdminNodeName* is not specified, the *NodeName* value is used.

Note: *AdminNodeName* must be a resolvable network host name. For more information, see the topic *GPFS node adapter interface names* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

You must provide a *NodeDesc* for each node to be added to the GPFS cluster.

-r *RemoteShellCommand*
Specifies the fully-qualified path name for the remote shell program to be used by GPFS. The default value is **/usr/bin/ssh**.

The remote shell command must adhere to the same syntax format as the `ssh` command, but may implement an alternate authentication mechanism.

-R *RemoteFileCopy*
Specifies the fully-qualified path name for the remote file copy program to be used by GPFS. The default value is **/usr/bin/scp**.

The remote copy command must adhere to the same syntax format as the `scp` command, but may implement an alternate authentication mechanism.

--use-sudo-wrapper [sudo-user *UserName*]
Causes the nodes in the cluster to call the ssh and scp sudo wrapper scripts as the remote shell program and the remote copy program. For more information, see the topic *Running IBM Spectrum Scale without remote root login* in the *IBM Spectrum Scale: Administration Guide*.

--sudo-user *UserName*
Specifies a non-root admin user ID to be used when sudo wrappers are enabled and a root-level background process calls an administration command directly instead of through **sudo**. The GPFS daemon that processes the administration command specifies this non-root user ID instead of the root ID when it needs to run internal commands on other nodes. For more information, see the topic *Root-level processes that call administration commands directly* in the *IBM Spectrum Scale: Administration Guide*.

To disable this feature, specify the key word **DELETE** instead of a user name, as in the following example:

```
mmchcluster --sudo-user DELETE
```

-C *ClusterName*
Specifies a name for the cluster. If the user-provided name contains dots then the command assumes that the user-provided name is a fully qualified domain name. Otherwise, to make the cluster name unique, the command appends the domain of a quorum node to the user-provided name. The maximum length of the cluster name including any appended domain name is 115 characters.

If the `-C` flag is omitted, the cluster name defaults to the name of a quorum node within the cluster definition.

-U *DomainName*
Specifies the UID domain name for the cluster.

-A
Specifies that GPFS daemons are to be automatically started when nodes come up. The default is not to start daemons automatically.

-c *ConfigFile*
Specifies a file containing GPFS configuration parameters with values different than the documented defaults. A sample file can be found in `/usr/lpp/mmfs/samples/mmfs.cfg.sample`. See the `mmchconfig` command for a detailed description of the different configuration parameters.

The `-c` ***ConfigFile*** parameter should be used only by experienced administrators. Use this file to set up only those parameters that appear in the `mmfs.cfg.sample` file. Changes to any other values may be ignored by GPFS. When in doubt, use the `mmchconfig` command instead.

--profile *ProfileName*
Specifies a predefined profile of attributes to be applied. System-defined profiles are located in /usr/lpp/mmfs/profiles/. All the configuration attributes listed under a cluster stanza will be changed as a result of this command.

The following system-defined profile names are accepted:

- `gpfsProtocolDefaults`
- `gpfsProtocolRandomIO`

A user's profiles must be installed in /var/mmfs/etc/. The profile file specifies GPFS configuration parameters with values different than the documented defaults. A user-defined profile must not begin with the string 'gpfs' and must have the .profile suffix.

User-defined profiles consist of the following stanzas:

```
%cluster:
[CommaSeparatedNodesOrNodeClasses:]ClusterConfigurationAttribute=Value
...
```

```
%filesystem:
FilesystemConfigurationAttribute=Value
```

See the **mmchconfig** command for a detailed description of the different configuration parameters. A sample file can be found in /usr/lpp/mmfs/samples/sample.profile.

Note: User-defined profiles should be used only by experienced administrators. When in doubt, use the **mmchconfig** command instead.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmcrcluster` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

To create a GPFS cluster made of all of the nodes that are listed in the file `/u/admin/nodelist`, issue the following command:

```
mmcrcluster -N /u/admin/nodelist
```

where the file `/u/admin/nodelist` has the following contents:

```
k164n04.kgn.ibm.com:quorum
k164n05.kgn.ibm.com:quorum
k164n06.kgn.ibm.com
```

The command displays output as the following example:

```
Mon May 10 10:59:09 EDT 2010: mmcrcluster:
 Processing node k164n04.kgn.ibm.com
Mon May 10 10:59:09 EDT 2010: mmcrcluster:
 Processing node k164n05.kgn.ibm.com
Mon May 10 10:59:09 EDT 2010: mmcrcluster:
 Processing node k164n06.kgn.ibm.com
mmcrcluster: Command successfully completed
mmcrcluster: Warning: Not all nodes have proper
 GPFS license designations.
    Use the mmchlicense command to designate
     licenses as needed.
```

To confirm the creation, issue the following command:

```
mmlscluster
```

The command displays information as in the following example:

```
GPFS cluster information
========================
  GPFS cluster name:         k164n05.kgn.ibm.com
  GPFS cluster id:           680681562214606028
  GPFS UID domain:           k164n05.kgn.ibm.com
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp

  Repository type: CCR
```

```
 Node  Daemon node name    IP address     Admin node name  Designation
 ------------------------------------------------------------------------
  1   k164n04.kgn.ibm.com  198.117.68.68  k164n04.kgn.ibm.com  quorum
  2   k164n05.kgn.ibm.com  198.117.68.71  k164n05.kgn.ibm.com  quorum
  3   k164n06.kgn.ibm.com  198.117.68.70  k164n06.kgn.ibm.com
```

See also

- “mmaddnode command” on page 34
- “mmchconfig command” on page 170
- “mmdelnode command” on page 375
- “mmlscluster command” on page 492
- “mmlsconfig command” on page 495

Location

/usr/lpp/mmfs/bin

mmcrfileset command

Creates a GPFS fileset.

Synopsis

```
mmcrfileset Device FilesetName [-p afmAttribute=Value...] [-t Comment]
   [--inode-space {new [--inode-limit MaxNumInodes[:NumInodesToPreallocate]] | ExistingFileset}]
   [--allow-permission-change PermissionChangeMode]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmcrfileset` command constructs a new fileset with the specified name. The new fileset is empty except for a root directory, and does not appear in the directory namespace until the `mmlinkfileset` command is issued. The `mmcrfileset` command is separate from the `mmlinkfileset` command to allow the administrator to establish policies and quotas on the fileset before it is linked into the namespace.

For information on filesets, see the *Filesets* section in the *IBM Spectrum Scale: Administration Guide*.

Parameters

Device
The device name of the file system to contain the new fileset.

File system names need not be fully-qualified. `fs0` is as acceptable as `/dev/fs0`.

FilesetName
Specifies the name of the fileset to be created.

Note the following restrictions on fileset names:

- The name must be unique within the file system.
- The length of the name must be in the range 1-255.
- The name `root` is reserved for the fileset of the root directory of the file system.
- The name cannot be the reserved word *new*. However, the character string `new` can appear within a fileset name.
- The name cannot begin with a hyphen (-).
- The name cannot contain the following characters: `/ ? $ & * ( ) ` # | [ ] \`
- The name cannot contain a white-space character such as blank space or tab.

-p *afmAttribute=Value*
Specifies an AFM configuration parameter and its value. More than one `-p` option can be specified.

The following AFM configuration parameter is required for the `mmcrfileset` command:

afmTarget
Identifies the home that is associated with the cache; specified in either of the following forms:

```
nfs://{Host|Map}/Target_Path
```

or

```
gpfs://[Map]/Target_Path
```

where:

nfs:// or gpfs://
Specifies the transport protocol.

Host*|*Map

Host
Specifies the server domain name system (DNS) name or IP address.

Map
Specifies the export map name. Information about Mapping is contained in the AFM Overview > Parallel data transfers section.

See the following examples:

1. An example of using the nfs:// protocol with a map name:

```
mmcrfileset fs3 singleWriter2 -p
        afmtarget=nfs://<map1>/gpfs/fs1/target1 -p afmmode=sw --inode-space new
```

2. An example of using the nfs:// protocol with a host name:

```
mmcrfileset fs3 singleWriter2 -p
        afmtarget=nfs://<hostname>/gpfs/fs1/target1 -p afmmode=sw --inode-space
new
```

3. An example of using the gpfs:// protocol without a map name:

```
mmcrfileset fs3 singleWriter1 -p
         afmtarget=gpfs:///gpfs/thefs1/target1 -p afmmode=sw --inode-space new
```

Note: If you are not specifying the map name, a '/' is still needed to indicate the path.

4. An example of using the gpfs:// protocol with a map name:

```
mmcrfileset fs3 singleWriter1 -p
        afmtarget=gpfs://<map>/gpfs/thefs1/target1 -p afmmode=sw --inode-space new
```

The following optional AFM configuration parameters are also valid:

afmAsyncDelay
Specifies (in seconds) the amount of time by which write operations are delayed (because write operations are asynchronous with respect to remote clusters). For write-intensive applications that keep writing to the same set of files, this delay is helpful because it replaces multiple writes to the home cluster with a single write containing the latest data. However, setting a very high value weakens the consistency of data on the remote cluster.

This configuration parameter is applicable only for writer caches (SW, IW, and primary), where data from cache is pushed to home.

Valid values are between 1 and 2147483647. The default is 15.

afmDirLookupRefreshInterval
Controls the frequency of data revalidations that are triggered by such lookup operations as `ls` or `stat` (specified in seconds). When a lookup operation is done on a directory, if the specified amount of time has passed, AFM sends a message to the home cluster to find out whether the metadata of that directory has been modified since the last time it was checked. If the time interval has not passed, AFM does not check the home cluster for updates to the metadata.

Valid values are 0 through 2147483647. The default is 60. In situations where home cluster data changes frequently, a value of 0 is recommended.

afmDirOpenRefreshInterval
Controls the frequency of data revalidations that are triggered by such I/O operations as `read` or `write` (specified in seconds). After a directory has been cached, `open` requests resulting from I/O operations on that object are directed to the cached directory until the specified amount of time has passed. Once the specified amount of time has passed, the `open` request gets directed to a gateway node rather than to the cached directory.

Valid values are between 0 and 2147483647. The default is 60. Setting a lower value guarantees a higher level of consistency.

afmEnableAutoEviction
Enables eviction on a given fileset. A `yes` value specifies that eviction is allowed on the fileset. A `no` value specifies that eviction is not allowed on the fileset.

See also the topic about cache eviction in the *IBM Spectrum Scale: Administration Guide*.

afmExpirationTimeout
Is used with `afmDisconnectTimeout` (which can be set only through `mmchconfig`) to control how long a network outage between the cache and home clusters can continue before the data in the cache is considered out of sync with home. After `afmDisconnectTimeout` expires, cached data remains available until `afmExpirationTimeout` expires, at which point the cached data is considered expired and cannot be read until a reconnect occurs.

Valid values are 0 through 2147483647. The default is disable.

afmFileLookupRefreshInterval
Controls the frequency of data revalidations that are triggered by such lookup operations as `ls` or `stat` (specified in seconds). When a lookup operation is done on a file, if the specified amount of time has passed, AFM sends a message to the home cluster to find out whether the metadata of the file has been modified since the last time it was checked. If the time interval has not passed, AFM does not check the home cluster for updates to the metadata.

Valid values are 0 through 2147483647. The default is 30. In situations where home cluster data changes frequently, a value of 0 is recommended.

afmGateway
Specifies the user-defined gateway node for an AFM or AFM DR fileset, that gets preference over internal hashing algorithm. If the specified gateway node is not available, then AFM internally assigns a gateway node from the available list to the fileset. **afmHashVersion** value must be already set as '5'.

Note: Ensure that the file system is upgraded to IBM Spectrum Scale 5.0.2 or later.

The following command is an example of setting a user-defined gateway node to an AFM or AFM DR fileset -

```
#mmcrfileset <FileSystem> <fileset> -p
afmMode=<afmMode>,afmGateway=<GatewayNode> --inode-space new -p
afmTarget=<Target>
```

afmMode
Specifies the mode in which the cache operates. Valid values are the following:

single-writer | sw
Specifies single-writer mode.

read-only | ro
Specifies read-only mode. (For `mmcrfileset`, this is the default value.)

local-updates | lu
Specifies local-updates mode.

independent-writer | iw
Specifies independent-writer mode.

Primary | drp
Specifies the primary mode for AFM asynchronous data replication.

Secondary | drs
Specifies the secondary mode for AFM asynchronous data replication.

Changing from single-writer/read-only modes to read-only/local-updates/single-writer is supported. When changing from read-only to single-writer, the read-only cache is up-to-date. When changing from single-writer to read-only, all requests from cache should have been played

at home. Changing from local-updates to read-only/local-updates/single-writer is restricted. A typical dataset is set up to include a single cache cluster in single-writer mode (which generates the data) and one or more cache clusters in local-updates or read-only mode. AFM single-writer/independent-writer filesets can be converted to primary. Primary/secondary filesets cannot be converted to AFM filesets.

In case of AFM asynchronous data replication, the `mmchfileset` command cannot be used to convert to primary from secondary. For detailed information, see *AFM-based Asynchronous Disaster Recovery (AFM DR)* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

For more information, see the topic about caching modes in the *IBM Spectrum Scale: Administration Guide* chapter about active file management.

afmNumFlushThreads
Defines the number of threads used on each gateway to synchronize updates to the home cluster. The default value is 4, which is sufficient for most installations. The current maximum value is 1024, which is too high for most installations; setting this parameter to such an extreme value should be avoided.

afmParallelReadChunkSize
Defines the minimum chunk size of the read that needs to be distributed among the gateway nodes during parallel reads. Values are interpreted in terms of bytes. The default value of this parameter is 128 MiB, and the valid range of values is 0 to 2147483647. It can be changed cluster wide with the `mmchconfig` command. It can be set at fileset level using `mmcrfileset` or `mmchfileset` commands.

afmParallelReadThreshold
Defines the threshold beyond which parallel reads become effective. Reads are split into chunks when file size exceeds this threshold value. Values are interpreted in terms of MiB. The default value is 1024 MiB. The valid range of values is 0 to 2147483647. It can be changed cluster wide with the `mmchconfig` command. It can be set at fileset level using `mmcrfileset` or `mmchfileset` commands.

afmParallelWriteChunkSize
Defines the minimum chunk size of the write that needs to be distributed among the gateway nodes during parallel writes. Values are interpreted in terms of bytes. The default value of this parameter is 128 MiB, and the valid range of values is 0 to 2147483647. It can be changed cluster wide with the `mmchconfig` command. It can be set at fileset level using `mmcrfileset` or `mmchfileset` commands.

afmParallelWriteThreshold
Defines the threshold beyond which parallel writes become effective. Writes are split into chunks when file size exceeds this threshold value. Values are interpreted in terms of MiB. The default value of this parameter is 1024 MiB, and the valid range of values is 0 to 2147483647. It can be changed cluster wide with the `mmchconfig` command. It can be set at fileset level using `mmcrfileset` or `mmchfileset` commands.

afmPrefetchThreshold
Controls partial file caching and prefetching. Valid values are the following:

0
Enables full file prefetching. This is useful for sequentially accessed files that are read in their entirety, such as image files, home directories, and development environments. The file will be prefetched after three blocks have been read into the cache.

1-99
Specifies the percentage of file size that must be cached before the entire file is prefetched. A large value is suitable for a file accessed either randomly or sequentially but partially, for which it might be useful to ingest the rest of the file when most of it has been accessed.

100
Disables full file prefetching. This value only fetches and caches data that is read by the application. This is useful for large random-access files, such as databases, that are either too

big to fit in the cache or are never expected to be read in their entirety. When all data blocks are accessed in the cache, the file is marked as cached.

0 is the default value.

For local-updates mode, the whole file is prefetched when the first update is made.

afmPrimaryId
Specifies the unique primary ID of the primary fileset for asynchronous data replication. This is used for connecting a secondary to a primary.

afmRPO
Specifies the recovery point objective (RPO) interval for an AFM DR fileset. This attribute is disabled by default. You can specify a value with the suffix `M` for minutes, `H` for hours, or `W` for weeks. For example, for 12 hours specify 12H. If you do not add a suffix, the value is assumed to be in minutes. The range of valid values is 720 minutes - 2147483647 minutes.

afmShowHomeSnapshot
Controls the visibility of the home snapshot directory in cache. For this to be visible in cache, this variable has to be set to `yes`, and the snapshot directory name in the cache and home cannot be the same.

yes
Specifies that the home snapshot link directory is visible.

no
Specifies that the home snapshot link directory is not visible.

For more information about the snapshot, see *Peer snapshot -psnap* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

-t *Comment*
Specifies an optional comment that appears in the output of the `mmlsfileset` command. This comment must be less than 256 characters in length.

--inode-space {new | *ExistingFileset*}
Specifies the type of fileset to create, which controls how inodes are allocated:

new
Creates an independent fileset and its own dedicated inode space.

ExistingFileset
Creates a dependent fileset that will share inode space with the specified *ExistingFileset*. The *ExistingFileset* can be `root` or any other independent fileset.

If `--inode-space` is not specified, a dependent fileset will be created in the root inode space.

--inode-limit *MaxNumInodes*[:*NumInodesToPreallocate*]
Specifies the inode limit for the new inode space. The *NumInodesToPreallocate* specifies an optional number of inodes to preallocate when the fileset is created. This option is valid only when creating an independent fileset with the `--inode-space new` parameter.

Note: Preallocated inodes cannot be deleted or moved to another independent fileset. It is recommended to avoid preallocating too many inodes because there can be both performance and memory allocation costs associated with such preallocations. In most cases, there is no need to preallocate inodes because GPFS dynamically allocates inodes as needed.

--allow-permission-change *PermissionChangeMode*
Specifies the new permission change mode. This mode controls how **chmod** and ACL operations are handled on objects in the fileset. Valid modes are as follows:

chmodOnly
Specifies that only the UNIX change mode operation (**chmod**) is allowed to change access permissions (ACL commands and API will not be accepted).

setAclOnly
Specifies that permissions can be changed using ACL commands and API only (**chmod** will not be accepted).

chmodAndSetAcl
Specifies that **chmod** and ACL operations are permitted. If the **chmod** command (or **setattr** file operation) is issued, the result depends on the type of ACL that was previously controlling access to the object:

- If the object had a Posix ACL, it will be modified accordingly.
- If the object had an NFSv4 ACL, it will be replaced by the given UNIX mode bits.

Note: This is the default setting when a fileset is created.

chmodAndUpdateAcl
Specifies that **chmod** and ACL operations are permitted. If **chmod** is issued, the ACL will be updated by privileges derived from UNIX mode bits.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmcrfileset` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. This example creates a fileset in file system `gpfs1`:

```
# mmcrfileset gpfs1 fset1
```

   A sample output is as follows:

```
Fileset fset1 created with id 1.
```

2. This example adds `fset2` in file system `gpfs1` with the comment "another fileset":

```
# mmcrfileset gpfs1 fset2 -t "another fileset"
```

   A sample output is as follows:

```
Fileset fset2 created with id 2.
```

   To confirm the change, issue the following command:

```
# mmlsfileset gpfs1 -L
```

   A sample output is as follows:

```
Filesets in file system 'gpfs1':
Name Id RootInode ParentId Created                  InodeSpace MaxInodes AllocInodes Comment
root  0         3       -- Mon Apr 12 16:31:05 2010    0         8001536     8001536 root
fileset
fset1 1     13568        0 Mon Apr 12 16:32:28 2010    0               0           0
fset2 2     13569        0 Mon Apr 12 16:32:28 2010    0               0           0 another
fileset
```

See also

- “mmchfileset command” on page 224
- “mmdelfileset command” on page 369
- “mmlinkfileset command” on page 485
- “mmlsfileset command” on page 501
- “mmunlinkfileset command” on page 735

Location

`/usr/lpp/mmfs/bin`

mmcrfs command

Creates a GPFS file system.

Synopsis

```
mmcrfs Device {"DiskDesc[;DiskDesc...]" | -F StanzaFile}
       [-A {yes | no | automount}] [-B BlockSize] [-D {posix | nfs4}]
       [-E {yes | no}] [-i InodeSize] [-j {cluster | scatter}]
       [-k {posix | nfs4 | all}] [-K {no | whenpossible | always}]
       [-L LogFileSize] [-m DefaultMetadataReplicas]
       [-M MaxMetadataReplicas] [-n NumNodes] [-Q {yes | no}]
       [-p afmAttribute=Value[,afmAttribute=Value...]...]
       [-r DefaultDataReplicas] [-R MaxDataReplicas]
       [-S {yes | no | relatime}] [-T Mountpoint] [-t DriveLetter]
       [-v {yes | no}] [-z {yes | no}] [--filesetdf | --nofilesetdf]
       [--inode-limit MaxNumInodes[:NumInodesToPreallocate]]
       [--log-replicas LogReplicas] [--metadata-block-size MetadataBlockSize]
       [--perfileset-quota | --noperfileset-quota]
       [--mount-priority Priority] [--version VersionString]
       [--write-cache-threshold HAWCThreshold]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmcrfs` command to create a GPFS file system. The first parameter must be *Device* and it must be followed by either *DiskDescList* or `-F` *StanzaFile*. You can mount a maximum of 256 file systems in an IBM Spectrum Scale cluster at any one time, including remote file systems.

The performance of a file system is affected by the values that you set for block size, replication, and the maximum number of files (number of inodes).

For information about block size see the descriptions in this help topic of the `-B` *BlockSize* parameter and the `--metadata-block-size` parameter. Here are some general facts from those descriptions:

- The block size, subblock size, and number of subblocks per block of a file system are set when the file system is created and cannot be changed later.
- All the data blocks in a file system have the same block size and the same subblock size. Data blocks and subblocks in the system storage pool and those in user storage pools have the same sizes. An example of a valid block size and subblock size is a 4 MiB block with an 8 KiB subblock.
- All the metadata blocks in a file system have the same block size and the same subblock size. The metadata blocks and subblocks are set to the same sizes as data blocks and subblocks, unless the `--metadata-block-size` parameter is specified.
- If the system storage pool contains only `metadataOnly` NSDs, the metadata block can be set to a different size than the data block size with the `--metadata-block-size` parameter.

  Note: This setting can result in a change in the data subblock size and in the number of subblocks in a data block. For an example, see the subsection "Subblocks" in the description of the `-B` parameter later in this help topic.
- The data blocks and metadata blocks must have the same number of subblocks, even when the data block size and the metadata block size are different.
- The number of subblocks per block is derived from the smallest block size of any storage pool in the file system, including the system metadata pool.
- The block size cannot exceed the value of the cluster attribute `maxblocksize`, which can be set by the `mmchconfig` command.

For more information, see the topic *Block size* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

For information about replication factors, see the descriptions of the `-m`, `-M`, `-r`, and `-R` parameters in this help topic.

For information about the maximum number of files (number of inodes), see the description of the `--inode-limit` parameter later in this help topic.

Results

Upon successful completion of the `mmcrfs` command, these tasks are completed on all the nodes of the cluster:

- The mount point directory is created.
- The file system is formatted.

In GPFS v3.4 and earlier, disk information for the `mmcrfs` command was specified with disk descriptors, which have the following format. The second, third, and sixth fields are reserved:

```
DiskName:::DiskUsage:FailureGroup::StoragePool:
```

For compatibility with earlier versions, the `mmcrfs` command still accepts the traditional disk descriptors, but their use is deprecated.

Parameters

Device
The device name of the file system to be created.

File system names need not be fully qualified. `fs0` is as acceptable as `/dev/fs0`. However, file system names must be unique within a GPFS cluster. Do not specify an existing entry in `/dev`.

This must be the first parameter.

"*DiskDesc*[;*DiskDesc*...]"
A descriptor for each disk to be included. Each descriptor is separated by a semicolon (;). The entire list must be enclosed in quotation marks (' or "). The use of disk descriptors is discouraged.

-F *StanzaFile*
Specifies a file that contains the NSD stanzas and pool stanzas for the disks that are to be added to the file system. NSD stanzas have the following format:

```
%nsd:
  nsd=NsdName
  usage={dataOnly | metadataOnly | dataAndMetadata | descOnly}
  failureGroup=FailureGroup
  pool=StoragePool
  servers=ServerList
  device=DiskName
  thinDiskType={no | nvme | scsi | auto}
```

where:

nsd=*NsdName*
Specifies the name of an NSD that was previously created by the `mmcrnsd` command. For a list of available disks, issue the `mmlsnsd -F` command. This clause is mandatory for the `mmcrfs` command.

usage={dataOnly | metadataOnly | dataAndMetadata | descOnly}
Specifies the type of data that is to be stored on the disk.

Note: For information about the *system storage pool*, *user storage pools*, and the optional *metadata system pool*, see the topic *Internal storage pools* in the *IBM Spectrum Scale: Administration Guide*.

dataAndMetadata
Indicates that the disk contains both data and metadata. This is the default for disks in the system pool.

dataOnly
Indicates that the disk contains data and does not contain metadata. This is the default for disks in storage pools other than the system pool.

metadataOnly
Indicates that the disk contains metadata and does not contain data.

descOnly
Indicates that the disk contains no data and no file metadata. IBM Spectrum Scale uses this type of disk primarily to keep a copy of the file system descriptor. It can also be used as a third failure group in certain disaster recovery configurations. For more information, see the topic *Synchronous mirroring utilizing GPFS replication* in the *IBM Spectrum Scale: Administration Guide*.

failureGroup=*FailureGroup*
Identifies the failure group to which the disk belongs. A failure group identifier can be a simple integer or a topology vector that consists of up to three comma-separated integers. The default is -1, which indicates that the disk has no point of failure in common with any other disk.

GPFS uses this information during data and metadata placement to ensure that no two replicas of the same block can become unavailable due to a single failure. All disks that are attached to the same NSD server or adapter must be placed in the same failure group.

If the file system is configured with data replication, all storage pools must have two failure groups to maintain proper protection of the data. Similarly, if metadata replication is in effect, the system storage pool must have two failure groups.

Disks that belong to storage pools in which write affinity is enabled can use topology vectors to identify failure domains in a shared-nothing cluster. Disks that belong to traditional storage pools must use simple integers to specify the failure group.

pool=*StoragePool*
Specifies the storage pool to which the disk is to be assigned. The default is the system storage pool. To specify the system storage pool explicitly, type `system`:

```
pool=system
```

Only the system storage pool can contain `metadataOnly`, `dataAndMetadata`, or `descOnly` disks. Disks in other storage pools must be `dataOnly`.

servers=*ServerList*
A comma-separated list of NSD server nodes. This clause is ignored by the `mmcrfs` command.

device=*DiskName*
The block device name of the underlying disk device. This clause is ignored by the `mmcrfs` command.

thinDiskType={no | nvme | scsi | auto}
Specifies the space reclaim disk type:

Note: By default the system pool cannot contain both regular disks and thin provisioned disks. If you want to include both types of disk in the system pool, contact IBM Service for more information.

no
The disk does not support space reclaim. This value is the default.

nvme
The disk is a TRIM capable NVMe device that supports the `mmreclaimspace` command.

scsi
The disk is a thin provisioned SCSI disk that supports the `mmreclaimspace` command.

auto
The type of the disk is either `nvme` or `scsi`. IBM Spectrum Scale will try to detect the actual disk type automatically. To avoid problems, you should replace `auto` with the correct disk type, `nvme` or `scsi`, as soon as you can.

Note: The space reclaim auto-detection is enhanced in IBM Spectrum Scale 5.0.5. Use the auto keyword after you upgrade the cluster to IBM Spectrum Scale 5.0.5 or later.

For more information, see the topic *IBM Spectrum Scale with data reduction storage devices* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Pool stanzas have the following format:

```
%pool:
  pool=StoragePoolName
  blockSize=BlockSize
  usage={dataOnly | metadataOnly | dataAndMetadata}
  layoutMap={scatter | cluster}
  allowWriteAffinity={yes | no}
  writeAffinityDepth={0 | 1 | 2}
  blockGroupFactor=BlockGroupFactor
```

where:

pool=*StoragePoolName*
Is the name of a storage pool.

blockSize=*BlockSize*
Specifies the block size of the disks in the storage pool.

usage={dataOnly | metadataOnly | dataAndMetadata}
Specifies the type of data to be stored in the storage pool:

dataAndMetadata
Indicates that the disks in the storage pool contain both data and metadata. This is the default for disks in the system pool.

dataOnly
Indicates that the disks contain data and do not contain metadata. This is the default for disks in storage pools other than the system pool.

metadataOnly
Indicates that the disks contain metadata and do not contain data.

layoutMap={scatter | cluster}
Specifies the block allocation map type. When allocating blocks for a given file, GPFS first uses a round-robin algorithm to spread the data across all disks in the storage pool. After a disk is selected, the location of the data block on the disk is determined by the block allocation map type. If `cluster` is specified, GPFS attempts to allocate blocks in clusters. Blocks that belong to a particular file are kept adjacent to each other within each cluster. If `scatter` is specified, the location of the block is chosen randomly.

The `cluster` allocation method may provide better disk performance for some disk subsystems in relatively small installations. The benefits of clustered block allocation diminish when the number of nodes in the cluster or the number of disks in a file system increases, or when the file system's free space becomes fragmented. The `cluster` allocation method is the default for GPFS clusters with eight or fewer nodes and for file systems with eight or fewer disks.

The `scatter` allocation method provides more consistent file system performance by averaging out performance variations due to block location (for many disk subsystems, the location of the data relative to the disk edge has a substantial effect on performance). This allocation method is appropriate in most cases and is the default for GPFS clusters with more than eight nodes or file systems with more than eight disks.

The block allocation map type cannot be changed after the storage pool has been created.

allowWriteAffinity={yes | no}
Indicates whether the File Placement Optimizer (FPO) feature is to be enabled for the storage pool. For more information on FPO, see the *File Placement Optimizer* section in the *IBM Spectrum Scale: Administration Guide*.

writeAffinityDepth={0 | 1 | 2}
Specifies the allocation policy to be used by the node writing the data.

A write affinity depth of 0 indicates that each replica is to be striped across the disks in a cyclical fashion with the restriction that no two disks are in the same failure group. By default, the unit of striping is a block; however, if the block group factor is specified in order to exploit chunks, the unit of striping is a chunk.

A write affinity depth of 1 indicates that the first copy is written to the writer node. The second copy is written to a different rack. The third copy is written to the same rack as the second copy, but on a different half (which can be composed of several nodes).

A write affinity depth of 2 indicates that the first copy is written to the writer node. The second copy is written to the same rack as the first copy, but on a different half (which can be composed of several nodes). The target node is determined by a hash value on the fileset ID of the file, or it is chosen randomly if the file does not belong to any fileset. The third copy is striped across the disks in a cyclical fashion with the restriction that no two disks are in the same failure group. The following conditions must be met while using a write affinity depth of 2 to get evenly allocated space in all disks:

1. The configuration in disk number, disk size, and node number for each rack must be similar.
2. The number of nodes must be the same in the bottom half and the top half of each rack.

This behavior can be altered on an individual file basis by using the `--write-affinity-failure-group` option of the `mmchattr` command.

This parameter is ignored if write affinity is disabled for the storage pool.

blockGroupFactor=*BlockGroupFactor*
Specifies how many file system blocks are laid out sequentially on disk to behave like a single large block. This option only works if `--allow-write-affinity` is set for the data pool. This applies only to a new data block layout; it does not migrate previously existing data blocks. For more information, see the topic *File placement optimizer* in the *IBM Spectrum Scale: Administration Guide*.

-A {yes | no | automount}
Indicates when the file system is to be mounted:

yes
When the GPFS daemon starts. This is the default.

no
The file system is mounted manually.

automount
On non-Windows nodes, when the file system is first accessed. On Windows nodes, when the GPFS daemon starts.

Note: IBM Spectrum Protect for Space Management does not support file systems with the `-A` option set to **`automount`**.

-B *BlockSize*
Specifies the size of data blocks in the file system. By default this parameter sets the block size and subblock size for all the data blocks and metadata blocks in the file system. This statement applies to all the data blocks and metadata blocks in the system storage pool and all the data blocks in user storage pools.

Note: You can specify a different size for metadata blocks and subblocks, and this setting can change the size and number of data subblocks. For more information, see the description of the `--metadata-block-size` parameter later in this help topic.

Specify the value of the -B parameter with the character K or M. For example, to set the block size to 4 MiB with an 8 KiB subblock, type "-B 4M". The following table shows the supported block sizes with their subblock size:

Table 21. Supported block sizes with subblock size

Supported block sizes with subblock size
64 KiB block with a 2 KiB subblock
128 KiB block with a 4 KiB subblock
256 KiB, 512 KiB, 1 MiB, 2 MiB, or 4 MiB block with 8 KiB subblock
8 MiB or 16 MiB block with a 16 KiB subblock

Attention:

- A data block size of 4 MiB provides good sequential performance, makes efficient use of disk space, and provides good performance for small files. It works well for the widest variety of workloads.
- For information about suggested block sizes for different types of I/O and for different workloads and configuration types, see the topic *Block size* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Default block size

If the -B parameter is not specified then the data blocks and the metadata blocks in the file system are set to a default block size and subblock size. If the format version of the file system is 5.0.0 or greater (and the value of the **maxblocksize** cluster attribute is greater than 4 MiB) then the default block size is 4 MiB with an 8 KiB subblock.

The default block size depends on the file system format version of the new file system, which is determined by the specified value or the default value of the **--version** parameter. For more information, see the description of the **--version** parameter later in this help topic.

- If the format version of the new file system is 5.0.0 or greater (available in IBM Spectrum Scale 5.0.x or later) then the following settings apply:
  - The default block size is 4 MiB or the value of the **maxblocksize** attribute of the cluster, whichever is smaller. For more information about the **maxblocksize** attribute, see “mmchconfig command” on page 170.

    Note: But if **maxblocksize** is less than 256 KiB, then you must explicitly set -B *BlockSize* to 64 KiB or 128 KiB.
  - The subblock size is set from the appropriate row of Table 21 on page 323.
- If the format version of the file system is earlier than 5.0.0 (earlier than IBM Spectrum Scale 5.0.0) then the following settings apply:
  - The default block size is 256 KiB.
  - The subblock size is 1/32 of the block size.

Subblocks

By default the data blocks and metadata blocks in a file system are set to the same block size and the same subblock size. The block size and subblock size are determined either by a setting from Table 21 on page 323 (if -B is specified) or by the default sizes (if -B is not specified).

However, if metadata blocks are set to a different size than data blocks by the --metadata-block-size parameter, which is described later in this help topic, the following steps are taken automatically to determine the subblock sizes for data blocks and metadata blocks:

1. Determine the number of subblocks. This step is necessary because data blocks and metadata blocks must have the same number of subblocks:

a. Choose the block type with the smaller block size (usually the metadata block).
b. Set the subblock size from the appropriate row in Table 21 on page 323.
c. Find the number of subblocks by dividing the block size by the subblock size. This value will be the number of subblocks for both data blocks and metadata blocks.

For example, suppose that initially the block sizes are set to 16 MiB for data blocks and 1 MiB for metadata blocks. The smaller block size is 1 MiB for metadata blocks. From Table 21 on page 323, the subblock size for a block size of 1 MiB is 8 KiB. Therefore the number of subblocks is (1 MiB / 8 KiB) or 128 subblocks. Thus the following settings are determined:

- For metadata blocks, from the table, the block size is 1 MiB and the metadata subblock size is 8 KiB.
- Both data blocks and metadata blocks must have 128 subblocks.

2. Determine the subblock size for the other block type (usually the data block) by dividing the block size by the number of subblocks from Step 1. Continuing the example from Step 1, a data block must have 128 subblocks. Therefore the subblock size for data blocks is (16 MiB / 128) or 128 KiB. Note that this is different than the standard subblock size of 16 KIB for a 16 MIB block.

-D {nfs4 | posix}
Specifies whether a deny-write open lock blocks write operations, as it is required to do by NFS V4. File systems supporting NFS V4 must have `-D nfs4` set. The option `-D posix` allows NFS writes even in the presence of a deny-write open lock. If you intend to export the file system using NFS V4 or Samba, you must use `-D nfs4`. For NFS V3 (or if the file system is not NFS exported at all) use `-D posix`. The default is `-D nfs4`.

-E {yes | no}
Specifies whether to report *exact* `mtime` values (`-E yes`), or to periodically update the `mtime` value for a file system (`-E no`). If it is more desirable to display exact modification times for a file system, specify or use the default `-E yes`.

-i *InodeSize*
Specifies the byte size of inodes. Supported inode sizes are 512, 1024, and 4096 bytes. The default is 4096.

-j {cluster | scatter}
Specifies the default block allocation map type to be used if `layoutMap` is not specified for a given storage pool.

-k {posix | nfs4 | all}
Specifies the type of authorization supported by the file system:

posix
Traditional GPFS ACLs only (NFS V4 and Windows ACLs are not allowed). Authorization controls are unchanged from earlier releases.

nfs4
Support for NFS V4 and Windows ACLs only. Users are not allowed to assign traditional GPFS ACLs to any file system objects (directories and individual files).

all
Any supported ACL type is permitted. This includes traditional GPFS (`posix`) and NFS V4 and Windows ACLs (`nfs4`).

The administrator is allowing a mixture of ACL types. For example, `fileA` might have a `posix` ACL, while `fileB` in the same file system may have an NFS V4 ACL, implying different access characteristics for each file depending on the ACL type that is currently assigned. The default is `-k all`.

Avoid specifying `nfs4` or `all` unless files are to be exported to NFS V4 or Samba clients, or the file system is mounted on Windows. NFS V4 and Windows ACLs affect file attributes (mode) and have access and authorization characteristics that are different from traditional GPFS ACLs.

-K {no | whenpossible | always}
Specifies whether strict replication is to be enforced:

no
Indicates that strict replication is not enforced. GPFS tries to create the needed number of replicas, but still returns EOK if it can allocate at least one replica.

whenpossible
Indicates that strict replication is enforced provided the disk configuration allows it. If the number of failure groups is insufficient, strict replication is not enforced. This is the default value.

always
Indicates that strict replication is enforced.

For more information, see the topic "Strict replication" in the *IBM Spectrum Scale: Problem Determination Guide*.

-L *LogFileSize*
Specifies the size of the internal log files. The *LogFileSize* must be a multiple of the metadata block size. The default log file size is 32 MiB in most cases. However, if the data block size (parameter **-B**) is less than 512 KiB or if the metadata block size (parameter **--metadata-block-size**) is less than 256 KiB, then the default log file size is either 4 MiB or the metadata block size, whichever is greater. The minimum size is 256 KiB and the maximum size is 1024 MiB. Specify this value with the K or M character, for example: 8M.

The default log size works well in most cases. An increased log file size is useful when the highly available write cache feature (parameter `--write-cache-threshold`) is enabled.

-m *DefaultMetadataReplicas*
Specifies the default number of copies of inodes, directories, and indirect blocks for a file. Valid values are 1, 2, and 3. This value cannot be greater than the value of *MaxMetadataReplicas*. The default is 1.

-M *MaxMetadataReplicas*
Specifies the default maximum number of copies of inodes, directories, and indirect blocks for a file. Valid values are 1, 2, and 3. This value cannot be less than the value of *DefaultMetadataReplicas*. The default is 2.

-n *NumNodes*
The estimated number of nodes that will mount the file system in the local cluster and all remote clusters. This is used as a best guess for the initial size of some file system data structures. The default is 32. This value can be changed after the file system has been created but it does not change the existing data structures. Only the newly created data structure is affected by the new value. For example, *new storage pool*.

When you create a GPFS file system, you might want to overestimate the number of nodes that will mount the file system. GPFS uses this information for creating data structures that are essential for achieving maximum parallelism in file system operations (For more information, see *GPFS architecture* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*). If you are sure there will never be more than 64 nodes, allow the default value to be applied. If you are planning to add nodes to your system, you should specify a number larger than the default.

-Q {yes | no}
Activates quotas automatically when the file system is mounted. The default is `-Q no`. Issue the `mmdefedquota` command to establish default quota values. Issue the `mmedquota` command to establish explicit quota values.

To activate GPFS quota management after the file system has been created:

1. Mount the file system.
2. To establish default quotas:
   a. Issue the `mmdefedquota` command to establish default quota values.
   b. Issue the `mmdefquotaon` command to activate default quotas.
3. To activate explicit quotas:

a. Issue the `mmedquota` command to activate quota values.

b. Issue the `mmquotaon` command to activate quota enforcement.

-r *DefaultDataReplicas*
Specifies the default number of copies of each data block for a file. Valid values are 1, 2, and 3. This value cannot be greater than the value of *MaxDataReplicas*. The default is 1.

-R *MaxDataReplicas*
Specifies the default maximum number of copies of data blocks for a file. Valid values are 1, 2, and 3. This value cannot be less than the value of *DefaultDataReplicas*. The default is 2.

-S {yes | no | relatime}
Controls how the file attribute **atime** is updated.

Note: The attribute **atime** is updated locally in memory, but the value is not visible to other nodes until after the file is closed. To get an accurate value of **atime**, an application must call subroutine **gpfs_stat** or **gpfs_fstat**.

yes
The **atime** attribute is not updated. The subroutines **gpfs_stat** and **gpfs_fstat** return the time that the file system was last mounted with `relatime=no`. For more information, see the topics "mmmount command" on page 545 and *Mount options specific to IBM Spectrum Scale* in the *IBM Spectrum Scale: Administration Guide*.

no
The **atime** attribute is updated whenever the file is read. This option is the default if the minimum release level (**minReleaseLevel**) of the cluster is less than 5.0.0 when the file system is created.

relatime
The **atime** attribute is updated whenever the file is read, but only if one of the following conditions is true:

- The current file access time (**atime**) is earlier than the file modification time (**mtime**).
- The current file access time (**atime**) is greater than the **atimeDeferredSeconds** attribute. For more information, see the topic *mmchconfig command* in the *IBM Spectrum Scale: Command and Programming Reference*.

This setting is the default if the minimum release level (**minReleaseLevel**) of the cluster is 5.0.0 or greater when the file system is created.

For more information, see the topic *atime values* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

-T *MountPoint*
Specifies the mount point directory of the GPFS file system. If it is not specified, the mount point is set to *DefaultMountDir*/*Device*. The default value for *DefaultMountDir* is `/gpfs`, but it can be changed with the `mmchconfig` command.

-t *DriveLetter*
Specifies the drive letter to use when the file system is mounted on Windows.

-v {yes | no}
Verifies that specified disks do not belong to an existing file system. The default is `-v yes`. Specify `-v no` only when you want to reuse disks that are no longer needed for an existing file system. If the command is interrupted for any reason, use `-v no` on the next invocation of the command.

Important: Using `-v no` on a disk that already belongs to a file system will corrupt that file system. This will not be noticed until the next time that file system is mounted.

-z {yes | no}
Enable or disable DMAPI on the file system. Turning this option on requires an external data management application such as IBM Spectrum Protect hierarchical storage management (HSM) before the file system can be mounted. The default is `-z no`.

For further information regarding DMAPI for GPFS, see *GPFS-specific DMAPI events* in the *IBM Spectrum Scale: Command and Programming Reference*.

Note: IBM Spectrum Protect for Space Management does not support file systems with the `-A` option set to **automount**.

--filesetdf
Specifies that when quotas are enforced for a fileset (other than the root fileset), the `df` command reports either quota limit and usage or inode space capacity and usage for the fileset and not for the total file system. This option affects the `df` command behavior only on Linux nodes.

The `df` command reports quota limit and quota usage if quota is enabled for the fileset. If quota is disabled and `filesetdf` is enabled in IBM Spectrum Scale 5.1.1 or later with file system version 5.1.1 or later, then the **df** command reports inode space capacity and inode usage at the independent fileset-level. However, the **df** command reports the block space at the file system-level because the block space is shared with the whole file system.

Note: If quota is enabled, then no behavior change for **df** command with **filesetdf** enabled, regardless of the cluster and file system versions.

--nofilesetdf
Specifies that the numbers reported by the `df` command are not based on the quotas for a fileset. The `df` command returns the numbers for the entire file system. This is the default.

--inode-limit ***MaxNumInodes*****[:*****NumInodesToPreallocate*****]**
Specifies the maximum number of files in the file system.

In a file system that does parallel file creates, the number of free inodes must be greater than 5% of the total number of inodes. If not, the performance of the file system can be degraded. To increase the number of inodes, issue the `mmchfs` command.

The parameter *NumInodesToPreallocate* specifies the number of inodes that the system immediately preallocates. If you do not specify a value for *NumInodesToPreallocate*, GPFS dynamically allocates inodes as needed.

You can specify the *NumInodes* and *NumInodesToPreallocate* values with a suffix, for example 100K or 2M. Note that in order to optimize file system operations, the number of inodes that are actually created may be greater than the specified value.

Note: Preallocated inodes created using the **mmcrfs** command are allocated only to the root fileset, and these inodes cannot be deleted or moved to another independent fileset. It is recommended to avoid preallocating too many inodes because there can be both performance and memory allocation costs associated with such preallocations. In most cases, there is no need to preallocate inodes because GPFS dynamically allocates inodes as needed.

--log-replicas ***LogReplicas***
Specifies the number of recovery log replicas. Valid values are 1, 2, 3, or DEFAULT. If not specified, or if DEFAULT is specified, the number of log replicas is the same as the number of metadata replicas currently in effect for the file system.

This option is applicable only if the recovery log is stored in the `system.log` storage pool. For more information about the `system.log` storage pool, see the topic *The system.log storage pool* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

--metadata-block-size ***MetadataBlockSize***
Sets the metadata block size. Setting the metadata block size to a smaller value than the data block size can improve file system performance, especially when the data block size is greater than 1 MiB.

By default the data blocks and metadata blocks in a file system are set to the same block size and the same subblock size. For more information about these settings see the description of the `-B` *BlockSize* parameter earlier in this help topic.

To set metadata blocks to a different size than data blocks, you must define a metadata-only system pool and specify the `--metadata-block-size` parameter when you issue the `mmcrfs` command. Follow these steps:

1. Define a pool stanza for a metadata-only system pool. Include the following settings:
   - Set **pool** to a valid pool name.

- Do not set a value for **blockSize**. The metadata block size is set in the `--metadata-block-size` parameter of the `mmcrfs` command.
- Set **usage** to `metadataOnly`.

2. Define an NSD stanza that includes the following settings:
   - Set **usage** to **metadataOnly**.
   - Set **pool** to the name of the pool stanza that you defined in Step 1.
3. Include the NSD stanza and the pool stanza in the stanza file that you will pass to the `mmcrfs` command.
4. When you run the `mmcrfs` command, include the `--metadata-block-size` parameter and specify a valid block size from Table 21 on page 323.

Note: When metadata blocks are set to a different size than data blocks, the subblock sizes are ultimately determined by an automatic sequence of steps in the `mmcrfs` command processing. For more information, see the "Subblocks" subtopic in the description of the `--B` *BlockSize* parameter earlier in this help topic.

--perfileset-quota
Sets the scope of user and group quota limit checks to the individual fileset level (rather than the entire file system).

`--noperfileset-quota`
Sets the scope of user and group quota limit checks to the entire file system (rather than per individual fileset). This is the default.

--mount-priority ***Priority***
Controls the order in which the individual file systems are mounted at daemon startup or when one of the `all` keywords is specified on the `mmmount` command.

File systems with higher *Priority* numbers are mounted after file systems with lower numbers. File systems that do not have mount priorities are mounted last. A value of zero indicates no priority. This is the default.

--version ***VersionString***
Specifies the file system format version of the new file system, such as `4.2.3.0`. A file system format version is associated with a file system format number (for example, 17.0) that determines the features that are enabled in the new file system. For more information about these values, see the topic *File system format changes between versions of IBM Spectrum Scale* in the *IBM Spectrum Scale: Administration Guide*.

If you do not specify this parameter, the file system format version of the new file system defaults to the version of IBM Spectrum Scale that is installed on the node where you issue the command. For example, if IBM Spectrum Scale 4.2.3 is installed on the node where you issue the command, then the default file system format version for the new file system is 4.2.3.0.

Whether you specify the file system format version or let it assume the default value, the file system format version must be in the range 4.1.1.0 - *mRL*, where *mRL* is the minimum release level of the cluster (**minReleaseLevel**).

The file system format version also affects the default value of the `-B` *BlockSize* parameter. For more information, see the description of that parameter earlier in this help topic.

Important:

- A remote node with an installed product version of IBM Spectrum Scale (for example, 4.2.3) that is less than the file system format version of the new file system (such as 5.0.0) will not be able to access the file system.
- Windows nodes can mount only file systems with a file system format version greater than or equal to 3.2.1.5.
- If you do not specify this parameter and the installed product version of the node where you issue the command is greater than the minimum release level (**minReleaseLevel**) of the cluster, then

the command returns with an error message and prompts you to upgrade the minimum release level. To avoid this result, specify a file system format version with the **--version** parameter.

- In many contexts you might want to let the file system format version assume its default value. However, specifying an explicit file system format version can be useful or necessary in the following contexts:
  - When nodes in the cluster are running different versions of IBM Spectrum Scale.
  - When you want to make the file system available to remote clusters in which nodes are running an earlier version of IBM Spectrum Scale.

--profile ***ProfileName***
Specifies a predefined profile of attributes to be applied. System-defined profiles are located in /usr/lpp/mmfs/profiles/. All the file system attributes listed under a file system stanza are changed as a result of this command. The following system-defined profile names are accepted:

- gpfsProtocolDefaults
- gpfsProtocolRandomIO

The file system attributes are applied at file system creation. If there is a current profile in place on the system (use mmlsconfig profile to check), then the file system is created with those attributes and values listed in the profile's file system stanza. The default is to use whatever attributes and values associate with the current profile setting.

Furthermore, any and all file system attributes from an installed profile file can be by-passed with '--profile=userDefinedProfile', where the userDefinedProfile is a profile file has been installed by the user in /var/mmfs/etc/.

User-defined profiles consist of the following stanzas:

```
%cluster:
[CommaSeparatedNodesOrNodeClasses:]ClusterConfigurationAttribute=Value
...
%filesystem:
FilesystemConfigurationAttribute=Value
...
```

A sample file can be found in `/usr/lpp/mmfs/samples/sample.profile`. See the **mmchconfig** command for a detailed description of the different configuration parameters.

User-defined profiles should be used only by experienced administrators. When in doubt, use the **mmchconfig** command instead.

--write-cache-threshold ***HAWCThreshold***
Specifies the maximum length (in bytes) of write requests that will be initially buffered in the highly-available write cache before being written back to primary storage. Only synchronous write requests are guaranteed to be buffered in this fashion.

A value of 0 disables this feature. 64K is the maximum supported value. Specify in multiples of 4K.

This feature can be enabled or disabled at any time (the file system does not need to be unmounted). For more information about this feature, see the topic *Highly-available write cache (HAWC)* in the *IBM Spectrum Scale: Administration Guide*.

-p ***afmAttribute***
Specifies the AFM parameters that to be set on the file system for the file system-level migration by using AFM. If you set AFM parameters while you are creating a new file system, it allows migration of data from a source file system to the newly created IBM Spectrum Scale file system by using AFM migration method. This method does not require creation of any AFM mode fileset. Instead, AFM will be enabled on the "root" fileset of the file system. The supported AFM mode that can be enabled on the file system are AFM LU mode and AFM RO mode. Conversion of the RO mode to the LU mode is permitted. After the migration is completed, you can disable AFM relationship by using the **mmchfileset Device root -p afmTarget=disable** command and later use this as a regular

file system. After AFM relationship is disabled, it cannot be enabled again. For more information, see *Migration from the legacy hardware by using AFM* in the Library and related publications.

AFM supports the following parameters for file system-level migration:

afmTarget
Identifies the home that is associated with the cache. The home is specified in either of the following forms:

```
nfs://{Host|Map}/Source_Path
```

Where:

nfs://
Specifies the transport protocol.

Source_Path
Specifies the export path.

Host
Specifies the server domain name system (DNS) name or IP address.

Map
Specifies the export map name. For more information about mapping, see *Parallel data transfers* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

The **afmTarget** parameter examples are as follows:

1. Use NFS protocol without mapping.

```
# mmcrfs Device ... -p afmTarget=<Host|IP>://Source_Path,afmmode=ro
```

2. Use NFS protocol with mapping.

```
# mmcrfs Device ... -p afmtarget=nfs://<map1>/Source_Path,afmmode=ro
```

afmMode
Specifies the AFM fileset mode. Valid values are as follows:

read-only | ro
Specifies the read-only mode. You can fetch data into the ro-mode fileset for read-only purpose.

local-updates | lu
Specifies the local-updates mode. You can fetch data into the lu-mode fileset and update it locally. The modified data will not be synchronized to the home and stays local.

Conversion of the ro mode to the lu mode is supported for file system-level migration. For more information, see *Caching modes* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

To disable the AFM relationship from the file system, complete the following steps:

1. Unmount the file system on all cluster nodes.
2. Disable the AFM relationship by issuing the following command:

```
# mmchfileset fs1 root -p afmMode=disable
```

A sample output is as follows:

```
Warning! Once disabled, AFM cannot be re-enabled on this fileset. Do you wish to
continue? (yes/no) yes

Warning! Fileset should be verified for uncached files and orphans. If already
verified, then skip this step. Do you wish to verify same? (yes/no) no

Fileset root changed.
```

afmDirLookupRefreshInterval

Controls the frequency of data revalidations that are triggered by lookup operations such as ls or stat (specified in seconds). When a lookup operation is performed on a directory, if the specified time passed, AFM sends a message to the home cluster to find out whether the metadata of that directory is modified since the last time it was checked. If the time interval did not pass, AFM does not check the home cluster for updates to the metadata.

Valid values are 0 – 2147483647. The default is 60. Where home cluster data changes frequently, value 0 is recommended.

afmDirOpenRefreshInterval

Controls the frequency of data revalidations that are triggered by such I/O operations as read or write (specified in seconds). After a directory is cached, open requests that are resulting from I/O operations on that object are directed to the cached directory until the specified amount of time has passed. After the specified time passed, the open request is directed to a gateway node rather than to the cached directory.

Valid values are 0 - 2147483647. The default is 60. Set a lower value for a higher level of consistency.

afmFileLookupRefreshInterval

Controls the frequency of data revalidations that are triggered by lookup operations such as ls or stat (specified in seconds). When a lookup operation is performed on a file, if the specified time passed, AFM sends a message to the home cluster to find out whether the metadata of the file is modified since the last time it was checked. If the time interval did not pass, AFM does not check the home cluster for updates to the metadata.

Valid values are 0 – 2147483647. The default is 30. Where home cluster data changes frequently, value 0 is recommended.

afmFileOpenRefreshInterval

Controls the frequency of data revalidations that are triggered by I/O operations such as read or write (specified in seconds). After a file is cached, open requests from I/O operations on that object are directed to the cached file until the specified time passed. After the specified time passed, the open request is directed to a gateway node rather than to the cached file.

Valid values are 0 – 2147483647. The default is 30. Set a lower value for a higher level of consistency.

afmParallelReadChunkSize

Defines the minimum chunk size of the read that needs to be distributed among the gateway nodes during parallel reads. Values are interpreted in bytes. The default value of this parameter is 128 MiB, and the valid range of values is 0 – 2147483647. It can be changed cluster wide with the **mmchconfig** command. It can be set at fileset level by using the **mmcrfileset** or **mmchfileset** commands.

afmParallelReadThreshold

Defines the threshold beyond which parallel reads become effective. Reads are split into chunks when file size exceeds this threshold value. Values are interpreted in MiB. The default value is 1024 MiB. The valid range of values is 0 – 2147483647. It can be changed cluster wide with the **mmchconfig** command. It can be set at fileset level by using **mmcrfileset** or **mmchfileset** commands.

Exit status

0

Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the `mmcrfs` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

This example shows how to create a file system named `gpfs1` using three disks, each with a block size of 512 KiB, allowing metadata and data replication to be 2, turning quotas on, and creating `/gpfs1` as the mount point. The NSD stanzas describing the three disks are assumed to have been placed in file `/tmp/freedisks`. To complete this task, issue the following command:

```
# mmcrfs gpfs1 -F /tmp/freedisks -B 512K -m 2 -r 2 -Q yes -T /gpfs1
```

A sample output is as follows:

```
The following disks of gpfs1 will be formatted on node c21f1n13:
    hd2n97: size 1951449088 KB
    hd3n97: size 1951449088 KB
    hd4n97: size 1951449088 KB
Formatting file system ...
Disks up to size 16 TB can be added to storage pool 'system'.
Creating Inode File
Creating Allocation Maps
Creating Log Files
Clearing Inode Allocation Map
Clearing Block Allocation Map
Formatting Allocation Map for storage pool 'system'
  19 % complete on Tue Feb 28 18:03:20 2012
  42 % complete on Tue Feb 28 18:03:25 2012
  62 % complete on Tue Feb 28 18:03:30 2012
  79 % complete on Tue Feb 28 18:03:35 2012
  96 % complete on Tue Feb 28 18:03:40 2012
 100 % complete on Tue Feb 28 18:03:41 2012
Completed creation of file system /dev/gpfs1.
mmcrfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

See also

- “mmchfs command” on page 232
- “mmdelfs command” on page 373
- “mmdf command” on page 387
- “mmedquota command” on page 404
- “mmfsck command” on page 410
- “mmlsfs command” on page 506
- “mmlspool command” on page 528

Location

`/usr/lpp/mmfs/bin`

mmcrnodeclass command

Creates user-defined node classes.

Synopsis

```
mmcrnodeclass ClassName -N {Node[,Node...] | NodeFile | NodeClass}
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmcrnodeclass` command to create user-defined node classes. After a node class is created, it can be specified as an argument on commands that accept the `-N` *NodeClass* option.

Parameters

ClassName
Specifies a name that uniquely identifies the user-defined node classes to create. An existing node name cannot be specified. Class names that end with `nodes` or system-defined node classes are reserved for use by GPFS.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies the nodes and node classes that will become members of the user-defined node class *ClassName*.

NodeClass cannot be a node class that already contains other node classes. For example, two user-defined node classes called `siteA` and `siteB` could be used to create a new node class called `siteAandB`, as follows:

```
mmcrnodeclass siteAandB -N siteA,siteB
```

The `siteAandB` node class cannot later be specified for *NodeClass* when creating new node classes.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmcrnodeclass` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

To create a user-defined node class called siteA that contains nodes c8f2c4vp1 and c8f2c4vp2, issue this command:

```
mmcrnodeclass siteA -N c8f2c4vp1,c8f2c4vp2
```

The system displays information similar to:

```
mmcrnodeclass: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

To display the member list of siteA, issue this command:

```
mmlsnodeclass siteA
```

The system displays information similar to:

```
Node Class Name       Members
--------------------- -----------------------------------------------------------
siteA                 c8f2c4vp1,c8f2c4vp2
```

See also

- “mmchnodeclass command” on page 251
- “mmdelnodeclass command” on page 379
- “mmlsnodeclass command” on page 520

Location

/usr/lpp/mmfs/bin

mmcrnsd command

Creates Network Shared Disks (NSDs) used by GPFS.

Synopsis

```
mmcrnsd -F StanzaFile [-v {yes | no}]
```

Availability

Available on all IBM Spectrum Scale editions.

Note: If the cluster is running the IBM Spectrum Scale Developer Edition, the `mmcrnsd` command calculates the total disk size for all the proposed new NSDs along with any previously created NSDs. If the total exceeds the license limit, the `mmcrnsd` command fails with an error message. For more information, see the “mmlslicense command” on page 511.

Description

The `mmcrnsd` command is used to create cluster-wide names for NSDs used by GPFS.

This is the first GPFS step in preparing disks for use by a GPFS file system. The input to this command consists of a file containing NSD stanzas describing the properties of the disks to be created. This file can be updated as necessary by the `mmcrnsd` command and can be supplied as input to the `mmcrfs`, `mmadddisk`, or `mmrpldisk` command.

The names that are created by the `mmcrnsd` command are necessary since disks connected to multiple nodes might have different disk device names on each node. The NSD names uniquely identify each disk. This command must be run for all disks that are to be used in GPFS file systems. The `mmcrnsd` command is also used to assign each disk an NSD server list that can be used for I/O operations on behalf of nodes that do not have direct access to the disk.

To identify that a disk has been processed by the `mmcrnsd` command, a unique NSD volume ID is written to the disk. All of the NSD commands (`mmcrnsd`, `mmlsnsd`, and `mmdelnsd`) use this unique NSD volume ID to identify and process NSDs.

After the NSDs are created, the GPFS cluster data is updated and they are available for use by GPFS.

Note: It is customary to use whole LUNs as NSDs. This generally provides the best performance and fault isolation. When SCSI-3 PR is in use, whole LUN use is required. In other deployment scenarios, it is possible to use disk partitions rather than whole LUNs, as long as care is taken to ensure that sharing of the same LUN through multiple partitions does not have an undesirable performance impact.

On Windows, GPFS will only create NSDs from empty disk drives. `mmcrnsd` accepts Windows *Basic* disks or *Unknown/Not Initialized* disks. It always re-initializes these disks so that they become *Basic GPT Disks* with a single *GPFS partition*. NSD data is stored in GPFS partitions. This allows other operating system components to recognize the disks are used. `mmdelnsd` deletes the partition tables that are created by `mmcrnsd`.

Results

Upon successful completion of the `mmcrnsd` command, these tasks are completed:

- NSDs are created.
- The *StanzaFile* contains NSD names to be used as input to the `mmcrfs`, `mmadddisk`, or the `mmrpldisk` commands.
- A unique NSD volume ID to identify each disk as an NSD has been written to the disk.
- An entry for each new disk is created in the GPFS cluster data.

Parameters

-F *StanzaFile*
Specifies the file containing the NSD stanzas for the disks to be created. NSD stanzas have this format:

```
%nsd: device=DiskName
  nsd=NsdName
  servers=ServerList
  usage={dataOnly | metadataOnly | dataAndMetadata | descOnly | localCache}
  failureGroup=FailureGroup
  pool=StoragePool
  thinDiskType={no | nvme | scsi | auto}
```

where:

device=*DiskName*
On UNIX, specifies the block device name that appears in `/dev` for the disk that you want to define as an NSD. Examples of disks that are accessible through a block device are SAN-attached disks.

Important: If a server node is specified, *DiskName* must be the `/dev` name for the disk device of the first listed NSD server node. If no server node is specified, *DiskName* must be the name of the disk device for the node from which the **mmcrnsd** command is issued.

On Windows, the disk number (for example, 3) of the disk you want to define as an NSD. Disk numbers appear in Windows Disk Management console and the DISKPART command line utility.

Important: If a server node is specified, *DiskName* must be the disk number from the first NSD server node that is defined in the server list. If no server node is specified, *DiskName* must be the name of the disk device for the node from which the **mmcrnsd** command is issued.

For the latest supported disk types, see IBM Spectrum Scale FAQ in IBM Documentation.

This clause is mandatory for the `mmcrnsd` command.

nsd=*NsdName*
Specifies the name of the NSD to be created. This name must not already be used as another GPFS disk name, and it must not begin with the reserved string 'gpfs'.

Note: This name can contain only the following characters: 'A' through 'Z', 'a' through 'z', '0' through '9', or '_' (the underscore). All other characters are not valid.

If you do not specify this clause, GPFS generates a unique name for the disk and adds the appropriate `nsd=`*NsdName* clause to the stanza file. The NSD is assigned a name according to the convention:

gpfs*NN*nsd
where *NN* is a unique nonnegative integer that is not used in any prior NSD.

servers=*ServerList*
Specifies a comma-separated list of NSD server nodes. You can specify up to eight NSD servers in this list. The defined NSD preferentially uses the first server on the list. If the first server is not available, the NSD uses the next available server on the list.

When you specify server nodes for your NSDs, the output of the `mmlscluster` command lists the host name and IP address combinations that are recognized by GPFS. The utilization of aliased host names that are not listed in the `mmlscluster` command output might produce undesired results.

There are two cases where a server list either must be omitted or is optional:

- For IBM Spectrum Scale RAID, a server list is not allowed. The servers are determined from the underlying vdisk definition. For more information about IBM Spectrum Scale RAID, see the *IBM Spectrum Scale RAID: Administration and Programming Reference*.
- For SAN configurations where the disks are SAN-attached to all nodes in the cluster, a server list is optional. However, if all nodes in the cluster do not have access to the disk, or if the file

system to which the disk belongs is to be accessed by other GPFS clusters, you must specify a server list.

usage={dataOnly | metadataOnly | dataAndMetadata | descOnly | localCache}
Specifies the type of data to be stored on the disk:

dataAndMetadata
Indicates that the disk contains both data and metadata. This is the default for disks in the system pool.

dataOnly
Indicates that the disk contains data and does not contain metadata. This is the default for disks in storage pools other than the system pool.

metadataOnly
Indicates that the disk contains metadata and does not contain data.

descOnly
Indicates that the disk contains no data and no file metadata. Such a disk is used solely to keep a copy of the file system descriptor, and can be used as a third failure group in certain disaster recovery configurations. For more information, see the topic *Synchronous mirroring utilizing GPFS replication* in the *IBM Spectrum Scale: Administration Guide*.

localCache
Indicates that the disk is to be used as a local read-only cache device.

This clause is ignored by the `mmcrnsd` command and is passed unchanged to the output file produced by the `mmcrnsd` command.

failureGroup=*FailureGroup*
Identifies the failure group to which the disk belongs. A failure group identifier can be a simple integer or a topology vector that consists of up to three comma-separated integers. The default is -1, which indicates that the disk has no point of failure in common with any other disk.

GPFS uses this information during data and metadata placement to ensure that no two replicas of the same block can become unavailable due to a single failure. All disks that are attached to the same NSD server or adapter must be placed in the same failure group.

If the file system is configured with data replication, all storage pools must have two failure groups to maintain proper protection of the data. Similarly, if metadata replication is in effect, the system storage pool must have two failure groups.

Disks that belong to storage pools in which write affinity is enabled can use topology vectors to identify failure domains in a shared-nothing cluster. Disks that belong to traditional storage pools must use simple integers to specify the failure group.

This clause is ignored by the `mmcrnsd` command, and is passed unchanged to the output file produced by the `mmcrnsd` command.

pool=*StoragePool*
Specifies the name of the storage pool to which the NSD is assigned. This clause is ignored by the `mmcrnsd` command and is passed unchanged to the output file produced by the `mmcrnsd` command.

The default value for `pool` is `system`.

thinDiskType={no | nvme | scsi | auto}
Specifies the space reclaim disk type:

no
The disk device supports space reclaim. This value is the default.

nvme
The disk is a TRIM capable NVMe device that supports the `mmreclaimspace` command.

scsi
The disk is a thin provisioned SCSI disk that supports the `mmreclaimspace` command.

auto
The type of the disk is either nvme or scsi. IBM Spectrum Scale will try to detect the actual disk type automatically. To avoid problems, you should replace auto with the correct disk type, nvme or scsi, as soon as you can.

Note: In 5.0.5, the space reclaim auto-detection is enhanced. It is encouraged to use the auto key-word after your cluster is upgraded to 5.0.5.

For more information, see the topic *IBM Spectrum Scale with data reduction storage devices* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

-v {yes | no}
Verify that the disks are not already formatted as an NSD.

A value of -v yes specifies that the NSDs are to be created only if each disk has not been formatted by a previous invocation of the mmcrnsd command, as indicated by the NSD volume ID on the disk. A value of -v no specifies that the disks are to be created irrespective of their previous state. The default is -v yes.

Important: Using -v no when a disk already belongs to a file system can corrupt that file system by making that physical disk undiscoverable by that file system. This will not be noticed until the next time that file system is mounted.

Upon successful completion of the mmcrnsd command, the *StanzaFile* file is rewritten to reflect changes made by the command, as follows:

- If an NSD stanza is found to be in error, the stanza is commented out.
- If an nsd=*NsdName* clause is not specified, and an NSD name is generated by GPFS, an nsd= clause will be inserted in the corresponding stanza.

You must have write access to the directory where the *StanzaFile* file is located in order to rewrite the created NSD information.

The disk usage, failure group, and storage pool specifications are preserved only if you use the rewritten file produced by the mmcrnsd command. If you do not use this file, you must either accept the default values or specify new values when creating NSD stanzas for other commands.

For backward compatibility, the mmcrnsd command will still accept the traditional disk descriptors, but their use is discouraged. For additional information about GPFS stanzas, see the help topic "Stanza files" in the *IBM Spectrum Scale: Administration Guide*.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the mmcrnsd command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

To create two new NSDs from the stanza file /tmp/newNSDstanza containing:

```
%nsd: device=/dev/sdav1
  servers=k145n05,k145n06
  failureGroup=4
```

```
%nsd:
  device=/dev/sdav2
  nsd=sd2pA
  servers=k145n06,k145n05
  usage=dataOnly
  failureGroup=5
  pool=poolA
```

Issue this command:

```
mmcrnsd -F /tmp/newNSDstanza
```

The output is similar to this:

```
mmcrnsd: Processing disk sdav1
mmcrnsd: Processing disk sdav2
mmcrnsd: 6027-1371 Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

As a result, two NSDs are created. The first one is assigned a name by the system, for example, gpfs1023nsd. The second disk is assigned the name sd2pA (as indicated by the nsd= clause in the stanza).

The newNSDstanza file is rewritten and looks like this (note the addition of an nsd= clause in the first stanza):

```
%nsd: nsd=gpfs1023nsd  device=/dev/sdav1
  servers=k145n05,k145n06
  failureGroup=4

%nsd:
  device=/dev/sdav2
  nsd=sd2pA
  servers=k145n06,k145n05
  usage=dataOnly
  failureGroup=5
  pool=poolA
```

See also

- “mmadddisk command” on page 28
- “mmchnsd command” on page 254
- “mmcrfs command” on page 318
- “mmdeldisk command” on page 364
- “mmdelnsd command” on page 381
- “mmlsnsd command” on page 522
- “mmrpldisk command” on page 690

Location

/usr/lpp/mmfs/bin

mmcrsnapshot command

Creates a snapshot of a file system or fileset at a single point in time.

Synopsis

```
mmcrsnapshot Device [[Fileset]:]Snapshot[,[[Fileset]:]Snapshot[-j
FilesetName[,FilesetName...]]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmcrsnapshot` command to create global snapshots or fileset snapshots at a single point in time. System data and existing snapshots are not copied. The snapshot function allows a backup or mirror program to run concurrently with user updates and still obtain a consistent copy of the file system as of the time the copy was created. Snapshots also provide an online backup capability that allows easy recovery from common problems such as accidental deletion of a file, and comparison with older versions of a file.

In IBM Spectrum Scale Release 4.2.1 and later, snapshot commands support the specification of multiple snapshots. Users can easily create and delete multiple snapshots. Also, system performance is increased by batching operations and reducing overhead.

In this release, the following new usages of the **mmcrsnapshot** command have been introduced:

```
# mmcrsnapshot device [[Fileset]:]Snapshot[,[[Fileset]:]Snapshot...]
```

```
# mmcrsnapshot device [[Fileset]:]Snapshot[,[[Fileset]:]Snapshot...] -j Fileset
```

```
# mmcrsnapshot device Snapshot -j Fileset1,Fileset2,...
```

A global snapshot is an exact copy of changed data in the active files and directories of a file system. Snapshots of a file system are read-only and they appear in a `.snapshots` directory located in the file system root directory. The files and attributes of the file system can be changed only in the active copy.

A fileset snapshot is an exact copy of changed data in the active files and directories of an independent fileset plus all dependent filesets. Fileset snapshots are read-only and they appear in a `.snapshots` directory located in the root directory of the fileset. The files and attributes of the fileset can be changed only in the active copy.

Snapshots may be deleted only by issuing the `mmdelsnapshot` command. The `.snapshots` directory cannot be deleted, though it can be renamed with the `mmsnapdir` command using the `-s` option.

Because global snapshots are not full, independent copies of the entire file system, they should not be used as protection against media failures. For information about protection against media failures, see the *Recoverability considerations* topic in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

For more information on global snapshots, see *Creating and maintaining snapshots of GPFS file systems* in the *IBM Spectrum Scale: Administration Guide*.

For more information on fileset snapshots, see *Fileset-level snapshots* in the *IBM Spectrum Scale: Administration Guide*.

Parameters

Device
The device name of the file system for which the snapshot is to be created. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

This must be the first parameter.

Fileset
The name of the fileset that contains the fileset snapshot to be created. If *Fileset* is not specified, the **mmcrsnapshot** command creates a global snapshot named Snapshot.

Note: Ensure that multiple snapshots and multiple filesets are not used together.

Snapshot
Specifies the name given to the snapshot.

The snapshot names are separated by a comma.

The snapshot specifier describes global and fileset snapshots. For example, *Fileset1*:*Snapshot1* specifies a fileset snapshot named Snapshot1 for fileset Fileset1. If *Fileset1* is empty, *Snapshot1* is a global snapshot named Snapshot1.

Each global snapshot name must be unique from any other global snapshots. If you do not want to traverse the root of the file system to access the global snapshot, a more convenient mechanism that enables a connection in each directory of the active file system can be enabled with the `-a` option of the `mmsnapdir` command.

Note: Ensure that the snapshot name is using the "@GMT-yyyy.MM.dd-HH.mm.ss" format in order to be identifiable by the Windows VSS.

For a fileset snapshot, *Snapshot* appears as a subdirectory of the `.snapshots` directory in the root directory of the fileset. Fileset snapshot names can be duplicated across different filesets. A fileset snapshot can also have the same name as a global snapshot. The `mmsnapdir` command provides an option to make global snapshots also available through the `.snapshots` in the root directory of all independent filesets.

Note:

- Ensure that the snapshot name does not include a colon (:) and a comma (,).
- Ensure that multiple snapshots and multiple filesets are not used together.

-j ***FilesetName***
Creates a snapshot that includes the specified fileset and all the dependent filesets that share the same inode space. *FilesetName* refers to an independent fileset. If `-j` is not specified, the `mmcrsnapshot` command creates a global snapshot that includes all filesets.

If `-j` is specified, fileset names can be considered as a list separated by commas.

Note:

- When a list of snapshots separated by a comma (,) is used with the `-j` option, the fileset is applicable to each snapshot that does not use the colon (:) syntax. The fileset name must not consist of a white space.
- Only one global snapshot and one snapshot can be specified in each fileset. Therefore, multiple snapshots must not be created with the `-j` option. IBM recommends that at the most one global snapshot and one fileset snapshot must be included in each independent fileset. If multiple snapshots are specified with the `-j` option, a snapshot with the same name is created for each fileset that has been listed.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmcrsnapshot` command when creating global snapshots.

Independent fileset owners can run the `mmcrsnapshot` command to create snapshots of the filesets they own.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To create a global snapshot `snap1`, for the file system `fs1`, issue the following command:

```
# mmcrsnapshot fs1 snap1
```

A sample output is as follows:

```
Writing dirty data to disk.
Quiescing all file system operations.
Writing dirty data to disk again.
Snapshot snap1 created with id 1.
```

Before issuing the command, the directory structure would appear similar to:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3
```

After the command has been issued, the directory structure would appear similar to:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3

/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3
```

If a second snapshot were to be created at a later time, the first snapshot would remain as is. Snapshots are made only of active file systems, not existing snapshots. For example:

```
# mmcrsnapshot fs1 snap2
```

A sample output is as follows:

```
Writing dirty data to disk.
Quiescing all file system operations.
Writing dirty data to disk again.
Snapshot snap2 created with id 2.
```

After the command has been issued, the directory structure would appear similar to:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3

/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3

/fs1/.snapshots/snap2/file1
```

```
/fs1/.snapshots/snap2/userA/file2
/fs1/.snapshots/snap2/userA/file3
```

2. To create a snapshot Snap3 of the fileset FsetF5-V2 for the file system fs1, issue the following command:

```
# mmcrsnapshot fs1 Snap3 -j FsetF5-V2
```

A sample output is as follows:

```
Writing dirty data to disk.
Quiescing all file system operations.
Writing dirty data to disk again.
Snapshot Snap3 created with id 69.
```

To display the snapshot that contains the FsetF5-V2 fileset, issue the following command:

```
# mmlssnapshot device -j FsetF5-V2
```

A sample output is as follows:

```
Snapshots in file system fs1:
Directory               SnapId    Status  Created                   Fileset
Snap3                   69        Valid   Wed Feb  1 12:55:51 2012  FsetF5-V2
```

3. To create a snapshot of the gpfs0 file system that can be viewed over SMB protocol with Windows VSS, issue the following command:

```
# mmcrsnapshot gpfs0 $(date --utc +@GMT-%Y.%m.%d-%H.%M.%S)
```

A sample output is as follows:

```
mmcrsnapshot gpfs0 $(date --utc +@GMT-%Y.%m.%d-%H.%M.%S)
Flushing dirty data for snapshot @GMT-2015.10.02-21.03.33...
Quiescing all file system operations.
Snapshot @GMT-2015.10.02-21.03.33 created with id 7.
```

4. To create a snapshot for the fset1, fset2, fset3 filesets for the file system fs1, run the following command:

```
# mmcrsnapshot fs1 snap1 -j fset1,fset2,fset3
```

A sample output is as follows:

```
Flushing dirty data for snapshot fset1:snap1 fset2:snap1 fset3:snap1 (1..3) of 3...
Quiescing all file system operations.
Snapshot fset1:snap1 created with id 1.
Snapshot fset2:snap1 created with id 2.
Snapshot fset3:snap1 created with id 3.
```

5. To specify different snapshot names for each fileset (fset1, fset2, and fset3) for the file system fs1, run the following command:

```
# mmcrsnapshot fs1 fset1:snapA,fset2:snapB,fset3:snapC
```

A sample output is as follows:

```
Flushing dirty data for snapshot fset1:snapA fset2:snapB fset3:snapC (1..3) of 3...
Quiescing all file system operations.
Snapshot fset1:snapA created with id 4.
Snapshot fset2:snapB created with id 5.
Snapshot fset3:snapC created with id 6.
```

To view the list of snapshots, run the following command:

```
# mmlssnapshot fs1
```

A sample output is as follows:

```
Snapshots in file system fs1:
Directory               SnapId    Status  Created                   Fileset
snap1                   1         Valid   Thu May 12 04:27:18 2016  fset1
snap1                   2         Valid   Thu May 12 04:27:18 2016  fset2
snap1                   3         Valid   Thu May 12 04:27:18 2016  fset3
snapA                   4         Valid   Thu May 12 04:28:33 2016  fset1
snapB                   5         Valid   Thu May 12 04:28:33 2016  fset2
snapC                   6         Valid   Thu May 12 04:28:33 2016  fset3
```

See also

- “mmdelsnapshot command” on page 383
- “mmlssnapshot command” on page 540
- “mmrestorefs command” on page 675
- “mmsnapdir command” on page 722

Location

/usr/lpp/mmfs/bin

mmdefedquota command

Sets default quota limits.

Synopsis

```
mmdefedquota {-u | -g | -j} Device
```

or

```
mmdefedquota {-u | -g} Device:Fileset
```

Availability

Available on all IBM Spectrum Scale editions. Available on AIX and Linux.

Description

Use the `mmdefedquota` command to set or change default quota limits. Default quota limits can be set for new users, groups, and filesets for a specified file system. Default quota limits can also be applied at a more granular level for new users and groups in a specified fileset.

Default quota limits can be set or changed only if the `-Q yes` option is in effect for the file system and if quotas are enabled with the `mmdefquotaon` command. To set default quotas at the fileset level, the `--perfileset-quota` option must also be in effect. If `--perfileset-quota` is in effect, all users and groups in the fileset `root` are not impacted by default quota unless they are explicitly set. The `-Q yes` and `--perfileset-quota` options are specified when you create a file system with the `mmcrfs` command or changing file system attributes with the `mmchfs` command. Use the `mmlsfs` command to display the current settings of these quota options.

The `mmdefedquota` command displays the current values for these limits, if any, and prompts you to enter new values in your default editor:

- The current block usage: The amount of disk space that is used by this user, group, or fileset, in 1 KB units; display only.
- The current inode usage: Display only.
- Inode soft limit.
- Inode hard limit.
- Block soft limit: The amount of disk space that this user, group, or fileset is allowed to use during normal operation.
- Block hard limit: The amount of disk space that this user, group, or fileset is allowed to use during the grace period.

  Note on block limits:

  – The command displays the current block limits in KB.

  – When you specify a block limit, you can add a suffix to the number to indicate the unit of measure: g, G, k, K, m, M, p, P, t, or T. If you do not specify a suffix, the command assumes that the number is in bytes.

  – The maximum block limit is 999999999999999 K (about 931322 T). For values greater than 976031318016 K (909 T) you must specify the equivalent value with the suffix K, M, or G or without any suffix.

  Note: A block or inode limit of 0 indicates no limit.

The `mmdefedquota` command waits for the edit window to be closed before it checks and applies new values. If an incorrect entry is made, reissue the command and enter the correct values.

When you set quota limits for a file system, consider replication within the file system. For more information, see the topic *Listing quotas* in the *IBM Spectrum Scale: Administration Guide*.

The EDITOR environment variable must contain a complete path name, for example:

```
export EDITOR=/bin/vi
```

Parameters

Device
The device name of the file system to have default quota values set for.

File system names need not be fully qualified.

Fileset
The name of a fileset in the file system to have default quota values set for.

Options

-g
Specifies that the default quota value is to be applied for new groups that access the specified file system or fileset.

-j
Specifies that the default quota value is to be applied for new filesets in the specified file system.

-u
Specifies that the default quota value is to be applied for new users that access the specified file system or fileset.

Note:

- The maximum files limit is 2147483647.
- See the *Note on block limits* earlier in this topic.
- If you want to display the current grace period, issue the command `mmrepquota -t`.

Exit status

0
Successful completion.

Nonzero
A failure has occurred.

Security

You must have root authority to run the `mmdefedquota` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

GPFS must be running on the node from which the `mmdefedquota` command is issued.

Examples

1. To set default quotas for new users of the file system `gpfs1`, issue the following command:

   ```
   mmdefedquota -u gpfs1
   ```

   The system displays the following information:

```
*** Edit quota limits for USR DEFAULT entry
NOTE: block limits will be rounded up to the next multiple of the block size.
      block units may be: K, M, G, T or P, inode units may be: K, M or G.
gpfs1: blocks in use: 0K, limits (soft = 0K, hard = 0K)
        inodes in use: 0, limits (soft = 0, hard = 0)
```

The following code block shows how to change the soft block limit to 19 GB, the hard block limit to 20 GB, the inode soft limit to 1 KB, and the inode hard limit to 20 KB:

```
*** Edit quota limits for USR DEFAULT entry
NOTE: block limits will be rounded up to the next multiple of the block size.
      block units may be: K, M, G, T or P, inode units may be: K, M or G.
gpfs1: blocks in use: 0K, limits (soft = 19G, hard = 20G)
        inodes in use: 0, limits (soft = 1K, hard = 20K)
```

After the edit window is closed, issue the following command to confirm the change:

```
mmlsquota -d -u gpfs1
```

The system displays the following information:

```
         Default Block Limits(KB)          |  Default File Limits
Filesystem type          quota      limit  |     quota    limit Remarks
gpfs1      USR        19922944   20971520  |      1024    20480
```

2. To set default quotas for new users of fileset `fset1` in file system `gpfs1`, issue the following command:

```
mmdefedquota -u gpfs1:fset1
```

The system displays the following information in your default editor:

```
*** Edit quota limits for USR DEFAULT entry for fileset fset1
NOTE: block limits will be rounded up to the next multiple of the block size.
      block units may be: K, M, G, T or P, inode units may be: K, M or G.
gpfs1: blocks in use: 0K, limits (soft = 0K, hard = 31457280K)
        inodes in use: 0, limits (soft = 0, hard = 0)
```

Change the soft block limit to 3 GB and the hard block limit to 6 GB, as shown in the following code block:

```
*** Edit quota limits for USR DEFAULT entry for fileset fset1
NOTE: block limits will be rounded up to the next multiple of the block size.
      block units may be: K, M, G, T or P, inode units may be: K, M or G.
gpfs1: blocks in use: 0K, limits (soft = 3G, hard = 6G)
        inodes in use: 0, limits (soft = 0, hard = 0)
```

After the edit window is closed, issue this command to confirm the change:

```
mmlsquota -d gpfs1:fset1
```

The system displays the following information:

```
         Default Block Limits(KB)                |  Default File Limits
Filesystem Fileset    type      quota      limit  |     quota    limit  entryType
gpfs1      fset1      USR     3145728    6291456  |         0        0  default on
gpfs1      fset1      GRP           0          0  |         0        0  default on
```

See also

- “mmchfs command” on page 232
- “mmcheckquota command” on page 220
- “mmcrfs command” on page 318
- “mmdefquotaoff command” on page 349
- “mmdefquotaon command” on page 352

- “mmedquota command” on page 404
- “mmlsfs command” on page 506
- “mmlsquota command” on page 535
- “mmquotaoff command” on page 651
- “mmrepquota command” on page 666

Location

`/usr/lpp/mmfs/bin`

mmdefquotaoff command

Deactivates default quota limit usage.

Synopsis

```
mmdefquotaoff [-u] [-g] [-j] [-v] [-d] {Device [Device...] | -a}
```

or

```
mmdefquotaoff [-u] [-g] [-v] [-d] {Device:Fileset ... | -a}
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmdefquotaoff` command deactivates default quota limits for file systems and filesets. If default quota limits are deactivated, new users, groups, or filesets will then have a default quota limit of 0, indicating no limit.

If none of the following options are specified, the `mmdefquotaoff` command deactivates all default quotas:

```
-u
-j
-g
```

If the `-a` option is not used, *Device* must be the last parameter specified.

Parameters

Device
The device name of the file system to have default quota values deactivated.

If more than one file system is listed, the names must be delimited by a space. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

Fileset
The name of a fileset in the file system to have default quota values deactivated.

Options

-a
Deactivates default quotas for all GPFS file systems in the cluster. When used in combination with the `-g` option, only group quotas are deactivated. When used in combination with the `-u` or `-j` options, only user or fileset quotas, respectively, are deactivated.

-d
Resets quota limits to zero for users, groups, or filesets.

When `--perfileset-quota` is not in effect for the file system, this option will reset quota limits to zero only for users, groups, or filesets that have default quotas established.

When `--perfileset-quota` is in effect for the file system, this option will reset quota limits to zero for users, groups, or filesets that have default quotas established only if *both* the file system and fileset-level default quotas are zero. If either file system or fileset-level default quotas exist, the default quotas will be switched to the level that is non-zero.

If this option is not chosen, existing quota entries remain in effect.

-g
Specifies that default quotas for groups are to be deactivated.

-j
Specifies that default quotas for filesets are to be deactivated.

-u
Specifies that default quotas for users are to be deactivated.

-v
Prints a message for each file system or fileset in which default quotas are deactivated.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmdefquotaoff` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

GPFS must be running on the node from which the `mmdefquotaoff` command is issued.

Examples

1. To deactivate default user quotas on file system `fs0`, issue this command:

```
mmdefquotaoff -u fs0
```

   To confirm the change, issue this command:

```
mmlsquota -d -u fs0
```

   The system displays information similar to:

```
         Default Block Limits(KB)          |  Default File Limits
Filesystem type          quota      limit  |     quota    limit Remarks
fs0        USR         no default limits
```

2. To deactivate default group quotas on all file systems, issue this command:

```
mmdefquotaoff -g -a
```

   To confirm the change, issue this command:

```
mmlsquota -d -g
```

   The system displays information similar to:

```
         Default Block Limits(KB)          |  Default File Limits
Filesystem type          quota      limit  |     quota    limit Remarks
fs0        GRP         no default limits

         Default Block Limits(KB)          |  Default File Limits
Filesystem type          quota      limit  |     quota    limit Remarks
fs1        GRP         no default limits
```

3. To deactivate both user and group default quotas for fileset `fset1` on file system `gpfs1`, issue this command:

```
mmdefquotaoff -d gpfs1:fset1
```

To confirm the change, issue this command:

```
mmlsquota -d gpfs1:fset1
```

The system displays information similar to:

```
         Default Block Limits(KB)          |  Default File Limits
Filesystem  Fileset  type   quota   limit  |     quota   limit  entryType
gpfs1       fset1    USR        0       0  |         0       0  default off
gpfs1       fset1    GRP        0       0  |         0       0  default off
```

See also

- “mmcheckquota command” on page 220
- “mmdefedquota command” on page 345
- “mmdefquotaon command” on page 352
- “mmedquota command” on page 404
- “mmlsquota command” on page 535
- “mmquotaoff command” on page 651
- “mmrepquota command” on page 666

Location

`/usr/lpp/mmfs/bin`

mmdefquotaon command

Activates default quota limit usage.

Synopsis

```
mmdefquotaon [-u] [-g] [-j] [-v] [-d] {Device [Device... ] | -a}
```

or

```
mmdefquotaon [-u] [-g] [-v] [-d] {Device:Fileset ... | -a}
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmdefquotaon` command activates default quota limits for file systems and filesets. If default quota limits are not applied, new users, groups, or filesets will have a quota limit of 0, indicating no limit.

To use default quotas, the `-Q yes` option must be in effect for the file system. To use default quotas at the fileset level, the `--perfileset-quota` option must also be in effect. The `-Q yes` and `--perfileset-quota` options are specified when creating a file system with the `mmcrfs` command or changing file system attributes with the `mmchfs` command. Use the `mmlsfs` command to display the current settings of these quota options.

If none of the following options are specified, the `mmdefquotaon` command activates all default quota limits:

```
-u
-j
-g
```

If the `-a` option is not used, *Device* must be the last parameter specified.

Default quotas are established for new users, groups of users or filesets by issuing the `mmdefedquota` command. Under the `-d` option, all users without an explicitly set quota limit will have a default quota limit assigned.

Parameters

Device
The device name of the file system to have default quota values activated.

If more than one file system is listed, the names must be delimited by a space. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

Fileset
The name of a fileset in the file system to have default quota values activated.

Options

-a
Activates default quotas for all GPFS file systems in the cluster. When used in combination with the `-g` option, only group quotas are activated. When used in combination with the `-u` or `-j` options, only user or fileset quotas, respectively, are activated.

-d
Assigns default quota limits to existing users, groups, or filesets when the `mmdefedquota` command is issued.

When `--perfileset-quota` is not in effect for the file system, this option will only affect existing users, groups, or filesets with no established quota limits.

When `--perfileset-quota` is in effect for the file system, this option will affect existing users, groups, or filesets with no established quota limits, and it will also change existing users or groups that refer to default quotas at the file system level into users or groups that refer to fileset-level default quota. For more information about default quota priorities, see the topic *Default quotas* in the *IBM Spectrum Scale: Administration Guide*.

If this option is not chosen, existing quota entries remain in effect and are not governed by the default quota rules.

-g
Specifies that default quotas for groups are to be activated.

-j
Specifies that default quotas for filesets are to be activated.

-u
Specifies that default quotas for users are to be activated.

-v
Prints a message for each file system or fileset in which default quotas are activated.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmdefquotaon` command.

The node on which you enter the command must be able to execute remote shell commands on any other administration node in the cluster. It must be able to do so without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

GPFS must be running on the node from which the `mmdefquotaon` command is issued.

Examples

1. To activate default user quotas on file system `fs0`, issue the following command:

```
# mmdefquotaon -u fs0
```

To confirm the change, issue the following command:

```
# mmlsfs fs0 -Q
```

A sample output is as follows:

```
flag value          description
---- -------------- ----------------------------------------
 -Q  user           Quotas enforced
     user           Default quotas enabled
```

2. To activate default group quotas on all file systems in the cluster, issue the following command:

```
# mmdefquotaon -g -a
```

To confirm the change, individually for each file system, issue the following command:

```
# mmlsfs fs1 -Q
```

A sample output is as follows:

```
flag value          description

---- -------------- ----------------------------
 -Q  group          Quotas enforced
     group          Default quotas enabled
```

3. To activate user, group, and fileset default quotas on file system `fs2`, issue the following command:

```
# mmdefquotaon fs2
```

To confirm the change, issue the following command:

```
# mmlsfs fs2 -Q
```

A sample output is as follows:

```
flag value               description
---- --------------      -------------------------
 -Q  user;group;fileset   Quotas enforced
     user;group;fileset   Default quotas enabled
```

4. To activate user default quota for fileset `fset1` on file system `gpfs1`, issue the following command:

```
# mmdefquotaon -d -u gpfs1:fset1
```

To confirm the change, issue the following command:

```
# mmlsquota -d gpfs1:fset1
```

A sample output is as follows:

```
            Default Block Limits(KB)   | Default File Limits
Filesystem Fileset type quota    limit | quota limit entryType
gpfs1      fset1   USR      0 31457280 |     0     0 default on
gpfs1      fset1   GRP      0        0 |     0     0 default off
```

In this example, notice the `entryType` for user quota displays `default on`. To also activate group default quota for `fset1` on file system `gpfs1`, issue the following command:

```
# mmdefquotaon -d -g gpfs1:fset1
```

To confirm the change, issue the following command:

```
# mmlsquota -d gpfs1:fset1
```

A sample output is as follows:

```
         Default Block Limits(KB)      | Default File Limits
Filesystem Fileset type quota    limit | quota limit entryType
gpfs1      fset1   USR      0 31457280 |     0     0 default on
gpfs1      fset1   GRP      0        0 |     0     0 default on
```

In this example, notice that the `entryType` for group quota also displays `default on` now.

See also

- “mmcheckquota command” on page 220
- “mmchfs command” on page 232

- “mmcrfs command” on page 318
- “mmdefedquota command” on page 345
- “mmdefquotaoff command” on page 349
- “mmedquota command” on page 404
- “mmlsfs command” on page 506
- “mmlsquota command” on page 535
- “mmquotaoff command” on page 651
- “mmrepquota command” on page 666

Location

`/usr/lpp/mmfs/bin`

mmdefragfs command

Reduces disk fragmentation by increasing the number of full free blocks available to the file system.

Synopsis

```
mmdefragfs Device [-i] [-u BlkUtilPct] [-P PoolName]
 [-N {Node[,Node...] | NodeFile | NodeClass}]
 [--pit-continues-on-error] [--qos QosClass]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmdefragfs` command to reduce fragmentation of a file system. The `mmdefragfs` command moves existing file system data within a disk to make more efficient use of disk blocks. The data is migrated to unused sub-blocks in partially allocated blocks, thereby increasing the number of free full blocks.

The `mmdefragfs` command can be run against a mounted or unmounted file system. However, best results are achieved when the file system is unmounted. When a file system is mounted, allocation status may change causing retries to find a suitable unused sub-block.

Note: On a file system that has a very low level of fragmentation, negative numbers can be seen in the output of `mmdefragfs` for free sub-blocks. This indicates that the block usage has in fact increased after running the `mmdefragfs` command. If negative numbers are seen, it does not indicate a problem and you do not need to rerun the `mmdefragfs` command.

Parameters

Device
The device name of the file system to have fragmentation reduced. File system names need not be fully-qualified. `fs0` is as acceptable as `/dev/fs0`.

This must be the first parameter.

-P *PoolName*
Specifies the pool name to use.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies the nodes that can be used in this disk defragmentation. This parameter supports all defined node classes. The default is `all` or the current value of the `defaultHelperNodes` parameter of the `mmchconfig` command.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

`--pit-continues-on-error`
Allows the **mmdefragfs** command to continue defragmenting the remaining files, if errors are encountered in the parallel inode traverse (PIT) phase that performs defragmentation of subblocks. An output file is generated if an error occurs in the PIT phase. The location of the file that logs the errors is displayed in the command output.

Note:

- The **mmdefragfs** command continues to run only if the error is reported in the PIT phase of the command execution. If the error is reported in other phases of command execution, the command stops running.

- The `--pit-continues-on-error` option works only if the minimum release level of the IBM Spectrum Scale cluster is 5.1.1 or later.

--qos *QosClass*
Specifies the Quality of Service for I/O operations (QoS) class to which the instance of the command is assigned. If you do not specify this parameter, the instance of the command is assigned by default to the `maintenance` QoS class. This parameter has no effect unless the QoS service is enabled. For more information, see the topic "mmchqos command" on page 263. Specify one of the following QoS classes:

`maintenance`
This QoS class is typically configured to have a smaller share of file system IOPS. Use this class for I/O-intensive, potentially long-running GPFS commands, so that they contribute less to reducing overall file system performance.

other
This QoS class is typically configured to have a larger share of file system IOPS. Use this class for administration commands that are not I/O-intensive.

For more information, see the topic *Setting the Quality of Service for I/O operations (QoS)* in *IBM Spectrum Scale: Administration Guide*.

Options

-i
Specifies to query the current disk fragmentation state of the file system. Does not perform the actual defragmentation of the disks in the file system.

-u *BlkUtilPct*
The average block utilization goal for the disks in the file system. The `mmdefragfs` command reduces the number of allocated blocks by increasing the percent utilization of the remaining blocks. The command automatically goes through multiple iterations until *BlkUtilPct* is achieved on all of the disks in the file system or until no progress is made in achieving *BlkUtilPct* from one iteration to the next, at which point it exits.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmdefragfs` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To query the fragmentation state of file system **fs0**, issue this command:

```
mmdefragfs fs0 -i
```

The system displays information similar to:

```
disk            disk size  in full  subblk in      %       %
name          in nSubblk   blocks   fragments free blk blk util
--------------- --------- --------- --------- --------  -------
```

```
nsd32              327680    277504     12931   84.688   96.054
nsd33              327680    315232       580   96.201   99.823
nsd21              327680    301824      2481   92.109   99.243
nsd34              327680    275904     13598   84.199   95.850
nsd30              327680    275808     13380   84.170   95.917
nsd19              327680    278496     12369   84.990   96.225
nsd31              327680    276224     12012   84.297   96.334
                 --------- --------- ---------           -------
(total)            2293760   2000992     67351            97.064
```

2. To reduce fragmentation of the file system **fs0** on all defined, accessible disks that are not stopped or suspended, issue this command:

```
mmdefragfs fs0
```

The system displays information similar to:

```
                    free subblk           free
  disk               in full            subblk in       %            %
  name               blocks    blk      franments     free blk     blk util
             before     after freed   before after before after before after
 ----------  ---------------------- ------------ ------------ ------------
  gpfs57nsd    28896     29888     31    1462   463  50.39 52.12  94.86 98.31
  gpfs60nsd    41728     43200     46    1834   362  59.49 61.59  93.55 98.66
               --------------------  --------------               -----------
 (total)      70624     73088    77     3296      825             93.63 98.84
```

3. To reduce fragmentation of all files in the **fs1** file system until the disks have 100% full block utilization, issue this command:

```
mmdefragfs fs1 -u 100
```

The system displays information similar to:

```
Defragmenting file system 'fs1'...

Defragmenting until full block utilization is 98.00%, currently 97.07%
 27.35 % complete on Tue May 26 14:25:42 2009    (    617882 inodes       4749 MB)
 82.65 % complete on Tue May 26 14:26:02 2009    (   1867101 inodes      10499 MB)
 89.56 % complete on Tue May 26 14:26:23 2009    (   2023206 inodes      14296 MB)
 90.01 % complete on Tue May 26 14:26:43 2009    (   2033337 inodes      17309 MB)
 90.28 % complete on Tue May 26 14:27:03 2009    (   2039551 inodes      19779 MB)
 91.17 % complete on Tue May 26 14:27:23 2009    (   2059629 inodes      23480 MB)
 91.67 % complete on Tue May 26 14:27:43 2009    (   2070865 inodes      26760 MB)
 92.51 % complete on Tue May 26 14:28:03 2009    (   2089804 inodes      29769 MB)
 93.12 % complete on Tue May 26 14:28:23 2009    (   2103697 inodes      32649 MB)
 93.39 % complete on Tue May 26 14:28:43 2009    (   2109629 inodes      34934 MB)
 95.47 % complete on Tue May 26 14:29:04 2009    (   2156805 inodes      36576 MB)
 95.66 % complete on Tue May 26 14:29:24 2009    (   2160915 inodes      38705 MB)
 95.84 % complete on Tue May 26 14:29:44 2009    (   2165146 inodes      40248 MB)
 96.58 % complete on Tue May 26 14:30:04 2009    (   2181719 inodes      41733 MB)
 96.77 % complete on Tue May 26 14:30:24 2009    (   2186053 inodes      43022 MB)
 96.99 % complete on Tue May 26 14:30:44 2009    (   2190955 inodes      43051 MB)
 97.20 % complete on Tue May 26 14:31:04 2009    (   2195726 inodes      43077 MB)
 97.40 % complete on Tue May 26 14:31:24 2009    (   2200378 inodes      43109 MB)
 97.62 % complete on Tue May 26 14:31:44 2009    (   2205201 inodes      43295 MB)
 97.83 % complete on Tue May 26 14:32:05 2009    (   2210003 inodes      43329 MB)
 97.85 % complete on Tue May 26 14:32:25 2009    (   2214741 inodes      43528 MB)
 97.86 % complete on Tue May 26 14:32:55 2009    (   2221888 inodes      43798 MB)
 97.87 % complete on Tue May 26 14:33:35 2009    (   2231453 inodes      44264 MB)
 97.88 % complete on Tue May 26 14:34:26 2009    (   2243181 inodes      45288 MB)
 100.00 % complete on Tue May 26 14:35:10 2009

               free subblk             free
disk            in full               subblk in         %            %
name            blocks       blk      fragments       free blk     blk util
             before    after freed   before    after before after before after
---------- ---------- ---------- ---------- ---------- ------------ ------------
nsd32        277504   287840   323    12931     2183  84.69 87.84  96.05 99.33
nsd33        315232   315456     7      580      185  96.20 96.27  99.82 99.94
nsd21        301824   303616    56     2481      666  92.11 92.66  99.24 99.80
nsd34        275904   285920   313    13598     3159  84.20 87.26  95.85 99.04
nsd30        275840   285856   313    13348     2923  84.18 87.24  95.93 99.11
nsd19        278592   288832   320    12273     1874  85.02 88.14  96.25 99.43
nsd31        276224   284608   262    12012     3146  84.30 86.86  96.33 99.04
           ---------- ---------- ----------                          ------------
(total)     2001120  2052128  1594    67223    14136                 97.07 99.38
```

```
Defragmentation complete, full block utilization is 99.04%.
```

See also

- “mmdf command” on page 387
- “mmrestripefs command” on page 682

Location

/usr/lpp/mmfs/bin

mmdelacl command

Deletes a GPFS access control list.

Synopsis

```
mmdelacl [-d] Filename
```

Availability

Available on all IBM Spectrum Scale editions. Available on AIX and Linux.

Description

Use the `mmdelacl` command to delete the extended entries of an access ACL of a file or directory, or to delete the default ACL of a directory.

Parameters

Filename
The path name of the file or directory for which the ACL is to be deleted. If the `-d` option is specified, *Filename* must contain the name of a directory.

Options

-d
Specifies that the default ACL of a directory is to be deleted.

Since there can be only one NFS V4 ACL (no separate default), specifying the `-d` flag for a file with an NFS V4 ACL is an error. Deleting an NFS V4 ACL necessarily removes both the ACL and any inheritable entries contained in it.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

The `mmdelacl` command may be issued only by the file or directory owner, the root user, or by someone with control (c) authority in the ACL for the file.

You may issue the `mmdelacl` command only from a node in the GPFS cluster where the file system is mounted.

Examples

To delete the default ACL for a directory named `project2`, issue this command:

```
mmdelacl -d project2
```

To confirm the deletion, issue this command:

```
mmgetacl -d project2
```

The system displays information similar to:

```
#owner:uno
#group:system
```

See also

- “mmeditacl command” on page 401
- “mmgetacl command” on page 428
- “mmputacl command” on page 617

Location

/usr/lpp/mmfs/bin

mmdelcallback command

Deletes one or more user-defined callbacks from the GPFS system.

Synopsis

```
mmdelcallback CallbackIdentifier[,CallbackIdentifier...]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmdelcallback` command to delete one or more user-defined callbacks from the GPFS system.

Parameters

CallbackIdentifier
Specifies a user-defined unique name that identifies the callback to be deleted. Use the `mmlscallback` command to see the name of the callbacks that can be deleted.

Note: Before you add or delete the **tiebreakerCheck** event, you must stop the GPFS daemon on all the nodes in the cluster.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmdelcallback` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

To delete the **test1** callback from the GPFS system, issue this command:

```
mmmdelcallback test1
```

The system displays information similar to:

```
mmdelcallback: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

See also

- "mmaddcallback command" on page 12
- "mmlscallback command" on page 490

Location

/usr/lpp/mmfs/bin

mmdeldisk command

Deletes disks from a GPFS file system.

Synopsis

```
mmdeldisk Device {"DiskName[;DiskName...]" | -F DescFile} [-a] [-c]
          [-m | -r | -b [--strict]] [-N {Node[,Node...] | NodeFile | NodeClass}]
          [--inode-criteria CriteriaFile] [-o InodeResultFile]
          [--pit-continues-on-error] [--qos QosClass]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmdeldisk` command migrates all data that would otherwise be lost to the remaining disks in the file system. It then removes the disks from the file system descriptor, preserves replication at all times, and optionally rebalances the file system after removing the disks.

The `mmdeldisk` command has the following two functions:

- Copying unreplicated data off the disks and removing references to the disks (`deldisk` step).
- Rereplicating or rebalancing blocks across the remaining disks (`restripe` step).

These two functions can be done in one pass over the file system, or in two passes if the `-a` option is specified.

Run the `mmdeldisk` command when system demand is low.

If a replacement for a failing disk is available, use the `mmrpldisk` command in order to keep the file system balanced. Otherwise, use one of these procedures to delete a disk:

- If the file system is replicated, replica copies can be preserved at all times by using the default `-r` option or the `-b` option.
- Using the `-m` option will not preserve replication during the `deldisk` step because it will only copy the minimal amount of data off the disk being deleted so that every block has at least one copy. Also, using the `-a` option will not preserve replication during the `deldisk` step, but will then re-establish replication during the subsequent `restripe` step.
- If you want to move all data off the disk before running `mmdeldisk`, use `mmchdisk` to suspend all the disks that will be deleted and run `mmrestripefs` with the `-r` or `-b` option. This step is no longer necessary, now that `mmdeldisk` does the same function. If `mmdeldisk` fails (or is canceled), it leaves the disks in the suspended state, and `mmdeldisk` can be retried when the problem that caused `mmdeldisk` to stop is corrected.

  If there are some disks marked as suspended or to be emptied and you run `mmdeldisk` command against any disk in the file system, **`mmrestripefs -r`** command is triggered by default. All the data from both the disks being deleted and suspended or to be emptied disks will be restriped. This slows down the execution of **`mmdeldisk`** command and triggers unexpected data movement. In order to avoid this, you should run **`mmchdisk fsName resume`** command to mark those suspended or to be emptied disks back to ready status before you run **`mmdeldisk`** command.
- If the disk is permanently damaged and the file system is not replicated, or if the `mmdeldisk` command repeatedly fails, see the *IBM Spectrum Scale: Problem Determination Guide* and search for *Disk media failure*.

If the last disk in a storage pool is deleted, the storage pool is deleted. The `mmdeldisk` command is not permitted to delete the `system` storage pool. A storage pool must be empty in order for it to be deleted.

Results

Upon successful completion of the `mmdeldisk` command, these tasks are completed:

- Data that has not been replicated from the target disks is migrated to other disks in the file system.
- Remaining disks are rebalanced, if specified.

Parameters

Device
The device name of the file system to delete the disks from. File system names need not be fully-qualified. `fs0` is as acceptable as `/dev/fs0`. This must be the first parameter.

"*DiskName*[;*DiskName*...]"
Specifies the names of the disks to be deleted from the file system. If there is more than one disk to be deleted, delimit each name with a semicolon (;) and enclose the list in quotation marks.

-F *DiskFile*
Specifies a file that contains the names of the disks (one name per line), to be deleted from the GPFS cluster.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies the nodes that participate in the restripe of the file system after the specified disks have been removed. This command supports all defined node classes. The default is `all` or the current value of the `defaultHelperNodes` parameter of the `mmchconfig` command.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

--inode-criteria *CriteriaFile*
Specifies the interesting inode criteria flag, where *CriteriaFile* contains a list of the following flags with one per line:

BROKEN
Indicates that a file has a data block with all of its replicas on disks that have been removed.

Note: BROKEN is always included in the list of flags even if it is not specified.

dataUpdateMiss
Indicates that at least one data block was not updated successfully on all replicas.

exposed
Indicates an inode with an exposed risk; that is, the file has data where all replicas are on suspended disks. This could cause data to be lost if the suspended disks have failed or been removed.

illCompressed
Indicates an inode in which file compression or decompression is deferred, or in which a compressed file is partly decompressed to allow the file to be written into or memory-mapped.

illPlaced
Indicates an inode with some data blocks that might be stored in an incorrect storage pool.

illReplicated
Indicates that the file has a data block that does not meet the setting for the replica.

metaUpdateMiss
Indicates that there is at least one metadata block that has not been successfully updated to all replicas.

unbalanced
Indicates that the file has a data block that is not well balanced across all the disks in all failure groups.

Note: If a file matches *any* of the specified interesting flags, all of its interesting flags (even those not specified) will be displayed.

-o ***InodeResultFile***
Contains a list of the inodes that met the interesting inode flags that were specified on the `--inode-criteria` parameter. The output file contains the following:

INODE_NUMBER
This is the inode number.

DISKADDR
Specifies a dummy address for later `tsfindinode` use.

SNAPSHOT_ID
This is the snapshot ID.

ISGLOBAL_SNAPSHOT
Indicates whether or not the inode is in a global snapshot. Files in the live file system are considered to be in a global snapshot.

INDEPENDENT_FSETID
Indicates the independent fileset to which the inode belongs.

MEMO (*INODE_FLAGS FILE_TYPE* [*ERROR*])
Indicates the inode flag and file type that will be printed:

`Inode flags`:

```
BROKEN
exposed
dataUpdateMiss
illCompressed
illPlaced
illReplicated
metaUpdateMiss
unbalanced
```

`File types`:

```
BLK_DEV
CHAR_DEV
DIRECTORY
FIFO
LINK
LOGFILE
REGULAR_FILE
RESERVED
SOCK
*UNLINKED*
*DELETED*
```

Notes:

1. An error message will be printed in the output file if an error is encountered when repairing the inode.
2. `DISKADDR`, `ISGLOBAL_SNAPSHOT`, and `FSET_ID` work with the `tsfindinode` tool (`/usr/lpp/mmfs/bin/tsfindinode`) to find the file name for each inode. `tsfindinode` uses the output file to retrieve the file name for each interesting inode.

--pit-continues-on-error
Allows the **mmdeldisk** command to continue repairing the remaining files, if errors are encountered in the parallel inode traverse (PIT) phase that performs scanning of user file metadata. An output file is generated if an error occurs in the PIT phase. The location of the file that logs the errors is displayed in the command output.

Note:

- The **mmdeldisk** command continues to run only if the error is reported in the PIT phase of the command execution. If the error is reported in other phases of command execution, the command stops running.
- The `--pit-continues-on-error` option works only if the minimum release level of the IBM Spectrum Scale cluster is 5.1.1 or later.

--qos *QosClass*
Specifies the Quality of Service for I/O operations (QoS) class to which the instance of the command is assigned. If you do not specify this parameter, the instance of the command is assigned by default to the `maintenance` QoS class. This parameter has no effect unless the QoS service is enabled. For more information, see the topic "mmchqos command" on page 263. Specify one of the following QoS classes:

`maintenance`
This QoS class is typically configured to have a smaller share of file system IOPS. Use this class for I/O-intensive, potentially long-running GPFS commands, so that they contribute less to reducing overall file system performance.

other
This QoS class is typically configured to have a larger share of file system IOPS. Use this class for administration commands that are not I/O-intensive.

For more information, see the topic *Setting the Quality of Service for I/O operations (QoS)* in *IBM Spectrum Scale: Administration Guide*.

Options

-a
Specifies that the `mmdeldisk` command *not* wait for rereplicating or rebalancing to complete before returning. When this flag is specified, the `mmdeldisk` command runs asynchronously and returns after the file system descriptor is updated and the rebalancing scan is started, but it does not wait for rebalancing to finish. If no rebalancing is requested (`-r` option is not specified), this option has no effect.

If `-m` is specified, this option has no effect. If `-r` or `-b` is specified (no option defaulting to `-r`), then the `deldisk` step is done using `-m`, and the `restripe` step is done using the specified option.

-b
Rebalances the file system to improve performance. Rebalancing removes file blocks from disks that are being deleted and attempts to distribute file blocks evenly across the remaining disks of the file system. In IBM Spectrum Scale 5.0.0 and later, rebalancing is implemented by a lenient round-robin method that typically runs faster than the previous method of strict round robin. To rebalance the file system using the strict round-robin method, include the `--strict` option that is described in the following text.

--strict
Rebalances the specified files with a strict round-robin method. In IBM Spectrum Scale 4.2.3 and earlier versions, rebalancing always uses this method.

Note: This option might result in much more data being moved than with the `-r` option.

Note: Rebalancing of files is an I/O intensive and time-consuming operation and is important only for file systems with large files that are mostly invariant. In many cases, normal file update and creation rebalances a file system over time without the cost of a complete rebalancing.

-c
Specifies that processing continues even in the event that unreadable data exists on the disks being deleted. Data that has not been replicated is lost. Replicated data is not lost as long as the disks containing the replication are accessible.

-m
Does minimal data copying to preserve any data that is located only on the disks being removed. This is the fastest way to get a disk out of the system, but it could reduce replication of some blocks of the files and metadata.

Note: This might be I/O intensive if there is a lot of data to be copied or re-replicated off the disks that are being deleted.

-r
Preserves replication of all files and metadata during the `mmdeldisk` operation (except when the `-a` option is specified). This is the default.

Note: This might be I/O intensive if there is a lot of data to be copied or re-replicated off the disks that are being deleted.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmdeldisk` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Example

To delete `gpfs1016nsd` from file system `fs1` and rebalance the files across the remaining disks, issue this command:

```
mmdeldisk fs1 gpfs1016nsd
```

The system displays information similar to:

```
Deleting disks ...
Scanning sp1 storage pool
Scanning user file metadata ...
 100.00 % complete on Tue Mar 13 15:48:51 2012
Scan completed successfully.
Checking Allocation Map for storage pool 'sp1'
tsdeldisk64 completed.
mmdeldisk: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

See also

- “mmadddisk command” on page 28
- “mmchdisk command” on page 212
- “mmlsdisk command” on page 497
- “mmrpldisk command” on page 690
- “mmrestripefs command” on page 682

Location

`/usr/lpp/mmfs/bin`

mmdelfileset command

Deletes a GPFS fileset.

Synopsis

```
mmdelfileset Device FilesetName [-f] [--pit-continues-on-error] [--qos QosClass]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmdelfileset` command deletes a GPFS fileset. When deleting a fileset, consider these points:

- The root fileset cannot be deleted.
- A fileset that is not empty cannot be deleted unless the `-f` flag is specified.

  Remember: If you use the `-f` flag, make sure that you remove all NFS exports defined on or inside the junction path of the fileset before you delete it. If you do not remove those exports before you delete the fileset, you might experience unexpected errors or issues when you create new filesets and exports.
- A fileset that is currently linked into the namespace cannot be deleted until it is unlinked with the `mmunlinkfileset` command.
- A dependent fileset can be deleted at any time.
- An independent fileset cannot be deleted if it has any dependent filesets or fileset snapshots.
- Deleting a dependent fileset that is included in a fileset or global snapshot removes it from the active file system, but it remains part of the file system in a deleted state.
- Deleting an independent fileset that is included in any global snapshots removes it from the active file system, but it remains part of the file system in a deleted state.
- A fileset in the deleted state is displayed in the `mmlsfileset` output with the fileset name in parenthesis. If the `-L` flag is specified, the latest including snapshot is also displayed. The `--deleted` option of the `mmlsfileset` command can be used to display only deleted filesets.
- The contents of a deleted fileset are still available in the snapshot, through some path name containing a `.snapshots` component, because it was saved when the snapshot was created.
- When the last snapshot that includes the fileset has been deleted, the fileset is fully removed from the file system.

For information on GPFS filesets, see *Information Lifecycle Management for GPFS* in *IBM Spectrum Scale: Administration Guide*.

Parameters

Device
The device name of the file system that contains the fileset.

File system names need not be fully-qualified. `fs0` is as acceptable as `/dev/fs0`.

FilesetName
Specifies the name of the fileset to be deleted.

-f
Forces the deletion of the fileset. All fileset contents are deleted. Any child filesets are first unlinked.

--pit-continues-on-error
Allows the **mmdelfileset** command to continue removing the remaining files, if errors are encountered in the parallel inode traverse (PIT) phase that performs the user file deletion. An output

file is generated if an error occurs during the user file deletion. The location of the file that logs the errors is displayed in the command output.

Note:

- The **mmdelfileset** command continues to run only if the error is reported in the PIT phase of the command execution. If the error is reported in other phases of command execution, the command stops running.
- The `--pit-continues-on-error` option works only if the minimum release level of the IBM Spectrum Scale cluster is 5.1.1 or later.

--qos *QosClass*
Specifies the Quality of Service for I/O operations (QoS) class to which the instance of the command is assigned. If you do not specify this parameter, the instance of the command is assigned by default to the `maintenance` QoS class. This parameter has no effect unless the QoS service is enabled. For more information, see the topic "mmchqos command" on page 263. Specify one of the following QoS classes:

maintenance
This QoS class is typically configured to have a smaller share of file system IOPS. Use this class for I/O-intensive, potentially long-running GPFS commands, so that they contribute less to reducing overall file system performance.

other
This QoS class is typically configured to have a larger share of file system IOPS. Use this class for administration commands that are not I/O-intensive.

For more information, see the topic *Setting the Quality of Service for I/O operations (QoS)* in *IBM Spectrum Scale: Administration Guide*.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmdelfileset` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. This sequence of commands illustrates what happens when attempting to delete a fileset that is linked.

   a. Command:

   ```
   mmlsfileset gpfs1
   ```

   The system displays output similar to:

   ```
   Filesets in file system 'gpfs1':
   Name                    Status      Path
   root                    Linked      /gpfs1
   fset1                   Linked      /gpfs1/fset1
   fset2                   Unlinked --
   ```

b. Command:

```
mmdelfileset gpfs1 fset1
```

The system displays output similar to:

```
Fileset fset1 must be unlinked to be deleted.
```

c. Command:

```
mmdelfileset gpfs1 fset2
```

The system displays output similar to:

```
Checking fileset ...
Checking fileset complete.
Deleting fileset ...
Fileset 'fset2' deleted.
```

d. To confirm the change, issue this command:

```
mmlsfileset gpfs1
```

The system displays output similar to:

```
Filesets in file system 'gpfs1':
Name                    Status    Path
root                    Linked    /gpfs1
fset1                   Linked    /gpfs1/fset1
```

2. This sequence of commands illustrates what happens when attempting to delete a fileset that contains user files.

a. Command:

```
mmlsfileset gpfs1
```

The system displays output similar to:

```
Filesets in file system 'gpfs1':
Name                    Status    Path
root                    Linked    /gpfs1
fset1                   Linked    /gpfs1/fset1
fset2                   Unlinked  --
```

b. Command:

```
mmdelfileset gpfs1 fset2
```

The system displays output similar to:

```
Fileset 'fset2' contains user files,
but can be deleted with the "-f" option.
```

c. Command:

```
mmdelfileset gpfs1 fset2 -f
```

The system displays output similar to:

```
Checking fileset ...
Checking fileset complete.
Deleting user files ...
100.00 % complete on Wed Feb 15 11:38:05 2012
Deleting fileset ...
Fileset 'fset2' deleted.
```

d. To confirm the change, issue this command:

```
mmlsfileset gpfs1
```

The system displays output similar to:

```
Filesets in file system 'gpfs1':
Name                    Status    Path
root                    Linked    /gpfs1
fset1                   Linked    /gpfs1/fset1
```

3. This command illustrates what happens when attempting to delete a fileset that contains user files fails, and the `--pit-continues-on-error` parameter is enabled.

```
mmdelfileset foofs fs1 --pit-continues-on-error -f
```

The system displays output similar to:

```
Checking fileset ...
   Checking fileset complete.
   Deleting user files ...
    100.00 % complete on Mon Feb 22 02:58:46 2021  (    100352 inodes with total        784
MB data processed)
   Check file '/var/mmfs/tmp/foofs.pit.interestingInodes.8589934595' on yuanzhcIO02 for
inodes with broken disk addresses or failures.
   mmdelfileset: Command failed. Examine previous error messages to determine cause.
```

See also

- “mmchfileset command” on page 224
- “mmcrfileset command” on page 311
- “mmlinkfileset command” on page 485
- “mmlsfileset command” on page 501
- “mmunlinkfileset command” on page 735

Location

`/usr/lpp/mmfs/bin`

mmdelfs command

Removes a GPFS file system.

Synopsis

```
mmdelfs Device [-p]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmdelfs` command removes all the structures for the specified file system from the nodes in the cluster.

Before you can delete a file system using the `mmdelfs` command, you must unmount it on all nodes.

Results

Upon successful completion of the `mmdelfs` command, these tasks are completed on all nodes:

- Deletes the character device entry from `/dev`.
- Removes the mount point directory where the file system had been mounted.

Parameters

Device
The device name of the file system to be removed. File system names need not be fully-qualified. `fs0` is as acceptable as `/dev/fs0`.

This must be the first parameter.

-p
Indicates that the disks are permanently damaged and the file system information should be removed from the GPFS cluster data even if the disks cannot be marked as **available**.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmdelfs` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

To delete file system `fs0`, issue this command:

```
mmdelfs fs0
```

The system displays information like the following:

```
GPFS: 6027-573 All data on the following disks of fs0 will be destroyed:
    gpfs9nsd
    gpfs10nsd
    gpfs15nsd
    gpfs17nsd
GPFS: 6027-574 Completed deletion of file system fs0.
mmdelfs: 6027-1371 Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

See also

- “mmcrfs command” on page 318
- “mmchfs command” on page 232
- “mmlsfs command” on page 506

Location

`/usr/lpp/mmfs/bin`

mmdelnode command

Removes one or more nodes from a GPFS cluster.

Synopsis

```
mmdelnode {-a | -N Node[,Node...] | NodeFile | NodeClass [--on-error-continue]}
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmdelnode` command to delete one or more nodes from the GPFS cluster. You may issue the `mmdelnode` command on any GPFS node.

A node cannot be deleted if any of the following are true:

1. It is a primary or secondary GPFS cluster configuration server.

   The node being deleted cannot be the primary or secondary GPFS cluster configuration server unless you intend to delete the entire cluster.

   You can determine whether a node is the primary or secondary configuration server by issuing the `mmlscluster` command. If the node is listed as of the servers and you still want to delete it without deleting the cluster, first use the `mmchcluster` command to assign another node as the server.

2. It is defined as an NSD server.

   The node being deleted cannot be defined as an NSD server for any disk unless you intend to delete the entire cluster.

   You can determine whether a node is an NSD server for one or more disks by issuing the `mmlsnsd` command. If the node is listed as an NSD server and you still want to delete it without deleting the cluster, first use the `mmchnsd` command to assign another node as an NSD server for the affected disks.

3. If the GPFS state is *unknown* and the node is reachable on the network.

   You cannot delete a node if both of the following are true:

   - The node responds to a TCP/IP ping command from another node.
   - The status of the node shows *unknown* when you use the `mmgetstate` command from another node in the cluster.

   Note: You will probably be able to delete such a node if you physically power it off.

4. If the node is configured as a performance monitoring collector. In such cases, you need to remove the node from the performance monitoring configuration by using the **mmperfmon config update --collectors** command before deleting the node. Deleting a collector node causes loss of all the collected perfmon stats data on the collector node.

5. If the node is defined as a Transparent cloud tiering node. You can determine whether a node is a Transparent cloud tiering node by issuing the **mmcloudgateway node list** command. If the node is listed as the Transparent cloud tiering node, and you still want to delete it without deleting the cluster, first use the **mmchnode** command to disable the Transparent cloud tiering node role.

   - If the node is a Transparent cloud tiering node, disable Transparent cloud tiering from the node by using the **mmchnode --cloud-gateway-disable** command, and then uninstall the Transparent cloud tiering rpms. Doing so ensures that the **mmdelnode** command does not fail on a Transparent cloud tiering node.

You must follow these rules when deleting nodes:

1. Unless all nodes in the cluster are being deleted, run the `mmdelnode` command from a node that will remain in the cluster.
2. Before you can delete a node, unmount all of the GPFS file systems and stop GPFS on the node to be deleted.
3. Exercise caution when shutting down GPFS on quorum nodes. If the number of remaining quorum nodes falls below the requirement for a quorum, you will be unable to perform file system operations. For more information, see *Quorum* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Each GPFS cluster is managed independently, so there is no automatic coordination and propagation of changes between clusters like there is between the nodes within a cluster. This means that if you permanently delete nodes that are being used as contact nodes by other GPFS clusters that can mount your file systems, you should notify the administrators of those GPFS clusters so that they can update their own environments.

Results

Upon successful completion of the `mmdelnode` command, the specified nodes are deleted from the GPFS cluster.

Parameters

-a
Delete all nodes in the cluster.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies the set of nodes to be deleted from the cluster.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

This command does not support a *NodeClass* of `mount`.

`--on-error-continue`

Enables the command to trap invalid nodes, remove those nodes from the node list, and continue to process the command. Without selecting this option, the command may exit for any reason when a node is found to be invalid.

Note: Not all error conditions relating to a node or the program function can allow the command to continue.

When this option is used, the *delete node* process runs in three stages and the progress is displayed on the console for each stage as it runs. The `--on-error-continue` option provides the following functions:

1. Some errors are trapped and the nodes that generated the errors are removed from the list of valid nodes to process. This allows the program to continue as long as there is at least one valid node remaining in the *delete node* process.
2. Presents the console output in more of a report format.
3. Creates three stages in the *delete node* process. The first two stages is where some node errors can be trapped and thus the node gets removed from the process. The third stage is more critical and will error out on any significant errors.
4. In stage one and two, if nodes failed to get removed from the cluster, two files are created at each stage to list the valid nodes and invalid nodes separately. The user gets informed of these files. This allows the user to rerun with just the valid node list from the last successful stage, after the reasons for failure are corrected.

Refer to the *Example* section for the example of the *delete node* process with the `--on-error-continue` option enabled.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the mmdelnode command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

You may issue the mmdelnode command from any node that will remain in the GPFS cluster.

Examples

1. To delete all of the nodes in the cluster, issue this command:

```
mmdelnode -a
```

   The system displays information similar to:

```
Verifying GPFS is stopped on all affected nodes ...
mmdelnode: Command successfully completed
mmdelnode: 6027-1371 Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

2. To delete nodes k145n12, k145n13, and k145n14, issue this command:

```
mmdelnode -N k145n12,k145n13,k145n14
```

   The system displays information similar to:

```
Verifying GPFS is stopped on all affected nodes ...
mmdelnode: Command successfully completed
mmdelnode: 6027-1371 Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

3. In the following example, the **mmdelnode** command is run with the --on-error-continue option. This option helps to continue the process of deleting a list of nodes from the cluster even if certain nodes in the list fail to get deleted from the cluster.

```
# mmdelnode -N notvalidhost.gpfs.net,c21m2n04.gpfs.net,c71f2u33p1.gpfs.net  --on-error-
continue
mmdelnode: On Error Continue option detected

=====================================================================
                          Stage 1
Running basic validation...
=====================================================================
---------------------------------------------------------------------
PROCESSING NODE: 1
Tue Feb 16 01:35:15 EST 2021: mmdelnode: Processing node notvalidhost.gpfs.net
mmdelnode: Incorrect node notvalidhost.gpfs.net specified for command.
mmdelnode: No nodes were found that matched the input specification.
---------------------------------------------------------------------
PROCESSING NODE: 2
Tue Feb 16 01:35:15 EST 2021: mmdelnode: Processing node c21m2n04.gpfs.net
---------------------------------------------------------------------
PROCESSING NODE: 3
Tue Feb 16 01:35:15 EST 2021: mmdelnode: Processing node c71f2u33p1.gpfs.net
---------------------------------------------------------------------
=====================================================================
                        Stage 1 Complete
Processed node count:   3
Number of valid nodes:  2
```

```
=============================================================

Stage 1 generated files:
      /tmp/mmdelnode_rpt__2021_02_16_01_35_15/valid_nodes_list
      /tmp/mmdelnode_rpt__2021_02_16_01_35_15/invalid_nodes_list

      If stage 2 fails, address any concerns, then rerun the command
      using the 'valid_nodes_list' file with the -N option to avoid
      processing invalid nodes again.
      Rerun the command with the invalid_nodes_list file with the -N
      option once you have addressed the concerns with those invalid
      nodes.

=============================================================
                          Stage 2
Running cluster config and feature validation checks...
=============================================================
-------------------------------------------------------------
PROCESSING NODE: 1
mmdelnode: : Processing node c21m2n04.gpfs.net
mmdelnode: Node c21m2n04.gpfs.net must be disabled as a CES node before trying to remove it
from the GPFS cluster.
mmdelnode: [E] Node c21m2n04.gpfs.net can not be deleted.
               It is marked for use by Transparent Cloud Tiering.
               To remove this node, first disable it with
               mmchnode --cloud-gateway-disable.

mmdelnode: The node c21m2n04.gpfs.net hit validation errors
           and was removed from the node deletion list.
-------------------------------------------------------------
PROCESSING NODE: 2
mmdelnode: : Processing node c71f2u33p1.gpfs.net
-------------------------------------------------------------
=============================================================
                       Stage 2 Complete
Processed node count:   2
Number of valid nodes:  1
=============================================================
Stage 2 generated files:
      /tmp/mmdelnode_rpt__2021_02_16_01_35_15/valid_nodes_list2
      /tmp/mmdelnode_rpt__2021_02_16_01_35_15/invalid_nodes_list2

      If stage 3 fails, address any concerns, then rerun the command
      using the 'valid_nodes_list2' file with the -N option to avoid
      processing invalid nodes again from Stage 2.
      Rerun the command with the invalid_nodes_list2 file with the -N
      option once you have addressed the concerns with those invalid
      nodes from Stage 2.


=============================================================
                          Stage 3
Running additional checks and committing node deletion updates.
This stage has critical checks.  It does not generate valid/invalid
node reports and will exit on any serious errors.
=============================================================
Verifying GPFS is stopped on all affected nodes ...
mmdelnode: Removing GPFS system files on all deleted nodes ...
mmdelnode: Command successfully completed
=============================================================
                      Stage 3 Complete
=============================================================
mmdelnode: Propagating the cluster configuration data to all affected nodes.
```

See also

- “mmaddnode command” on page 34
- “mmcrcluster command” on page 306
- “mmchconfig command” on page 170
- “mmlsfs command” on page 506
- “mmlscluster command” on page 492

Location

/usr/lpp/mmfs/bin

mmdelnodeclass command

Deletes user-defined node classes.

Synopsis

```
mmdelnodeclass ClassName[,ClassName...]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmdelnodeclass` command to delete existing user-defined node classes.

Parameters

ClassName
Specifies an existing user-defined node class to delete.

If *ClassName* was used to change configuration attributes with `mmchconfig`, and the configuration attributes are still referencing *ClassName*, then *ClassName* cannot be deleted. Use the `mmchconfig` command to remove the references to *ClassName* before deleting this user-defined node class.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmdelnodeclass` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

To display the current user-defined node classes, issue this command:

```
mmlsnodeclass --user
```

The system displays information similar to:

```
Node Class Name       Members
--------------------- -----------------------------------------------------------
siteA                 c6f1c3vp1,c6f1c3vp2
siteB                 c6f1c3vp4,c6f1c3vp5
```

To delete the `siteA` node class, issue this command:

```
mmdelnodeclass siteA
```

The system displays information similar to:

```
mmdelnodeclass: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

To display the updated list of user-defined node classes, issue this command:

```
mmlsnodeclass --user
```

The system displays information similar to:

```
Node Class Name       Members
--------------------- -----------------------------------------------------------
siteB                 c6f1c3vp4,c6f1c3vp5
```

See also

- “mmcrnodeclass command” on page 333
- “mmchnodeclass command” on page 251
- “mmlsnodeclass command” on page 520

Location

/usr/lpp/mmfs/bin

mmdelnsd command

Deletes Network Shared Disks (NSDs) from the GPFS cluster.

Synopsis

```
mmdelnsd {"DiskName[;DiskName...]" | -F DiskFile}
```

or

```
mmdelnsd -p NSDId [-N Node[,Node...]]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmdelnsd` command serves two purposes:

1. To delete NSDs from the GPFS cluster.
2. To remove the unique NSD volume ID left on a disk after the failure of a previous invocation of the `mmdelnsd` command. The NSD had been successfully deleted from the GPFS cluster but there was a failure to clear the unique volume ID from the disk.

NSDs being deleted cannot be part of any file system. To determine if an NSD belongs to a file system or not, issue the **mmlsnsd -d DiskName** command. If an NSD belongs to a file system, either the **mmdeldisk** or the **mmdelfs** command must be issued prior to deleting the NSDs from the GPFS cluster.

NSDs being deleted cannot be tiebreaker disks. To list the tiebreaker disks, issue the **mmlsconfig tiebreakerDisks** command. Use the **mmchconfig** command to assign new tiebreaker disks prior to deleting NSDs from the cluster. For information on tiebreaker disks, see *Quorum* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Results

Upon successful completion of the `mmdelnsd` command, these tasks are completed:

- All references to the disks are removed from the GPFS cluster data.
- Each disk is cleared of its unique NSD volume ID.
- On Windows, the disk's GPT partition table is removed leaving the disk Unknown/Not Initialized.

Parameters

***DiskName*[;*DiskName*...]**
: Specifies the names of the NSDs to be deleted from the GPFS cluster. Specify the names generated when the NSDs were created. Use the `mmlsnsd -F` command to display disk names. If there is more than one disk to be deleted, delimit each name with a semicolon (;) and enclose the list of disk names in quotation marks.

-F *DiskFile*
: Specifies a file containing the names of the NSDs, one per line, to be deleted from the GPFS cluster.

-N *Node*[,*Node*...]
: Specifies the nodes to which the disk is attached. If no nodes are listed, the disk is assumed to be directly attached to the local node.

  For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

-p *NSDId*
Specifies the NSD volume ID of an NSD that needs to be cleared from the disk as indicated by the failure of a previous invocation of the `mmdelnsd` command.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmdelnsd` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To delete `gpfs47nsd` from the GPFS cluster, issue this command:

   ```
   mmdelnsd "gpfs47nsd"
   ```

   The system displays output similar to:

   ```
   mmdelnsd: Processing disk gpfs47nsd
   mmdelnsd: 6027-1371 Propagating the cluster configuration data to all
     affected nodes.  This is an asynchronous process.
   ```

2. If after running **mmdelnsd** to delete an NSD, you experience a failure, the disk was not found. Run **mmdelnsd -p** *NSD Volume ID*. For example:

   ```
   mmdelnsd -p C0A8910B626630E
   ```

   This will remove the NSD definition from the GPFS configuration even if the NSD ID is not removed from the physical disk because it has been permanently lost.

See also

- “mmcrnsd command” on page 335
- “mmlsnsd command” on page 522

Location

`/usr/lpp/mmfs/bin`

mmdelsnapshot command

Deletes a GPFS snapshot.

Synopsis

```
mmdelsnapshot Device [[Fileset]:]Snapshot[,[[Fileset]:]Snapshot...
[-j FilesetName[,FilesetName...]][-N{all | mount | Node[,Node...]|NodeFile | NodeClass
[--pit-continues-on-error] [--qos QosClass]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmdelsnapshot` command to delete a GPFS snapshot.

Once the command is issued, the snapshot is marked for deletion and cannot be recovered.

If the node from which the `mmdelsnapshot` command is issued or the file system manager node fails, the snapshot might not be completely deleted. The `mmlssnapshot` command shows these snapshots with status `DeleteRequired`. Reissue the `mmdelsnapshot` from another node to complete the removal, or allow the snapshot to be cleaned up automatically by a later `mmdelsnapshot` command. A snapshot in this state cannot be accessed.

Any files open in the snapshot are forcibly closed. The user receives an `errno` of ESTALE on the next file access.

If a snapshot has file clones, you must delete the file clones or split them from their clone parents before you delete the snapshot. Use the `mmclone split` or `mmclone redirect` command to split file clones. Use a regular delete (`rm`) command to delete a file clone. If a snapshot is deleted that contains a clone parent, any attempts to read a block that refers to the missing snapshot returns an error. A policy file can be created to help determine whether a snapshot has file clones. See the *IBM Spectrum Scale: Administration Guide* for more information about file clones and policy files.

In IBM Spectrum Scale 4.2.1 and later, snapshot commands support the specification of multiple snapshots. Users can easily delete multiple snapshots for maintenance and cleanup operations. Also, system performance is increased by batching operations and reducing overhead.

In this release, the following new usages of the **mmdelsnapshot** command have been introduced:

```
mmdelsnapshot fs [[Fileset]:]Snapshot[,[[Fileset]:]Snapshot...]
```

```
mmdelsnapshot fs [[Fileset]:]Snapshot[,[[Fileset]:]Snapshot...] -j Fileset
```

```
mmdelnapshot fs Snapshot -j Fileset1,Fileset2,...
```

Parameters

Device
The device name of the file system for which the snapshot is to be deleted. File system names do not need to be fully qualified.

Fileset
Specifies the name of the fileset that contains the fileset snapshot to be deleted. If *Fileset* is not specified, the **mmdelsnapshot** command deletes a global snapshot named *Snapshot*.

Note: Ensure that multiple snapshots and multiple filesets are not used together.

Snapshot
Specifies the name of the snapshot to be deleted.

The snapshot names are separated by a comma.

The snapshot specifier describes global and fileset snapshots. For example, *Fileset1*:*Snapshot1* specifies a fileset snapshot named Snapshot1 for fileset Fileset1. If *Fileset1* is empty, *Snapshot1* is a global snapshot named Snapshot1.

Note: Ensure that the snapshot name does not include a colon (:), a comma (,), and whitespaces. If the snapshot name consists of a whitespace, the whitespace must be quoted and part of the snapshot name. For example, if the snap1 and snap A snapshots must be deleted, use the command **'mmdelsnapshot pk2 "snap A, snap1"'**.

-j ***FilesetName***
Specifies the name of the fileset that contains the fileset snapshot to be deleted (*SnapshotName*). If `-j` is not specified, the `mmdelsnapshot` command attempts to delete a global snapshot named *SnapshotName*.

Note: When a list of snapshots separated by a comma (,) is used with the `-j` option, the fileset is applicable to each snapshot that does not use the colon (:) syntax. The fileset name must not consist of a white space.

-N{all | mount | *Node*[,*Node*...]|NodeFile | NodeClass
Specifies the nodes that participate in deleting the snapshot. This command supports all defined node classes. The default is `all` or the current value of the `defaultHelperNodes` parameter of the `mmchconfig` command.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

--pit-continues-on-error
Allows the **mmdelsnapshot** command to continue removing the remaining files, if errors are encountered in the parallel inode traverse (PIT) phase that performs the deletion of files in the snapshot. An output file is generated if an error occurs while deleting files in the snapshot. The location of the file that logs the errors is displayed in the command output.

Note:

- The **mmdelsnapshot** command continues to run only if the error is reported in the PIT phase of the command execution. If the error is reported in other phases of command execution, the command stops running.
- The `--pit-continues-on-error` option works only if the minimum release level of the IBM Spectrum Scale cluster is 5.1.1 or later.

--qos ***QosClass***
Specifies the Quality of Service for I/O operations (QoS) class to which the instance of the command is assigned. If you do not specify this parameter, the instance of the command is assigned by default to the `maintenance` QoS class. This parameter has no effect unless the QoS service is enabled. For more information, see the topic "mmchqos command" on page 263. Specify one of the following QoS classes:

maintenance
This QoS class is typically configured to have a smaller share of file system IOPS. Use this class for I/O-intensive, potentially long-running GPFS commands, so that they contribute less to reducing overall file system performance.

other
This QoS class is typically configured to have a larger share of file system IOPS. Use this class for administration commands that are not I/O-intensive.

For more information, see the topic *Setting the Quality of Service for I/O operations (QoS)* in *IBM Spectrum Scale: Administration Guide*.

Exit status

0
Successful completion.

Nonzero
A failure occurred.

Security

You must have root authority to run the `mmdelsnapshot` command when you delete global snapshots.

Independent fileset owners can run the `mmdelsnapshot` command to delete snapshots of filesets that they own.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To delete the snapshot `snap1` for the file system `fs1`, run the following command:

```
mmdelsnapshot fs1 snap1
```

The output is similar to the following example:

```
Invalidating snapshot files...
Deleting snapshot files...
 100.00 % complete on Tue Feb 28 10:40:59 2012
Delete snapshot snap1 complete, err = 0
```

Before you issue the command, the directory might have the following structure:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3

/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3
```

After you issue the command, the directory has the following structure:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3
/fs1/.snapshots
```

2. To delete the snap1 snapshot from the fset1, fset2, and fset3 filesets for the file system `fs1`, run the following command:

```
mmdelsnapshot fs1 snap1 -j fset1,fset2,fset3
```

The system displays the following output:

```
Invalidating snapshot files in fset1:snap1 fset2:snap1 fset3:snap1 (1..3) of 3...
Deleting files in snapshot fset1:snap1...
 100.00 % complete on Thu May 12 04:43:07 2016  ( 100032 inodes with total 0 MB data
processed)
Invalidating snapshot files in fset1:snap1/F/...
Deleting files in snapshot fset2:snap1...
 100.00 % complete on Thu May 12 04:43:07 2016  ( 100032 inodes with total 0 MB data
processed)
Invalidating snapshot files in fset2:snap1/F/...
Deleting files in snapshot fset3:snap1...
 100.00 % complete on Thu May 12 04:43:07 2016  ( 100032 inodes with total 0 MB data
processed)
Invalidating snapshot files in fset3:snap1/F/...
```

```
Delete snapshot fset1:snap1 successful.
Delete snapshot fset2:snap1 successful.
Delete snapshot fset3:snap1 successful.
```

3. To specify the snapshot names that must be deleted from each fileset in the file system `fs1`, run the following command:

```
mmdelsnapshot fs1 fset1:snapA,fset2:snapB,fset3:snapC
```

The system displays the following command:

```
Invalidating snapshot files in fset3:snapC fset1:snapA fset2:snapB (1..3) of 3...
Deleting files in snapshot fset1:snapA...
 100.00 % complete on Thu May 12 04:44:46 2016  (  100032 inodes with total  0 MB data
processed)
Invalidating snapshot files in fset1:snapA/F/...
Deleting files in snapshot fset2:snapB...
 100.00 % complete on Thu May 12 04:44:46 2016  (  100032 inodes with total  0 MB data
processed)
Invalidating snapshot files in fset2:snapB/F/...
Deleting files in snapshot fset3:snapC...
 100.00 % complete on Thu May 12 04:44:46 2016  (  100032 inodes with total  0 MB data
processed)
Invalidating snapshot files in fset3:snapC/F/...
Delete snapshot fset1:snapA successful.
Delete snapshot fset2:snapB successful.
Delete snapshot fset3:snapC successful.
```

See also

- “mmclone command” on page 274
- “mmcrsnapshot command” on page 340
- “mmlssnapshot command” on page 540
- “mmrestorefs command” on page 675
- “mmsnapdir command” on page 722

Location

`/usr/lpp/mmfs/bin`

mmdf command

Queries available file space on a GPFS file system.

Synopsis

```
mmdf Device [-d] [-F] [-m] [-P PoolName] [-Y |--block-size {BlockSize | auto}]
  [--qos QOSClass]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmdf` command to display available file space on a GPFS file system. For each disk in the GPFS file system, the `mmdf` command displays this information, by failure group and storage pool:

- The size of the disk.
- The failure group of the disk.
- Whether the disk is used to hold data, metadata, or both.
- Available space in full blocks.
- Available space in subblocks ("fragments")

Displayed values are rounded down to a multiple of 1024 bytes. If the subblock ("fragment") size used by the file system is not a multiple of 1024 bytes, then the displayed values may be lower than the actual values. This can result in the display of a total value that exceeds the sum of the rounded values displayed for individual disks. The individual values are accurate if the subblock size is a multiple of 1024 bytes.

For the file system, the `mmdf` command displays the total number of inodes and the number available.

The `mmdf` command may be run against a mounted or unmounted file system.

Notes:

1. This command is I/O intensive and should be run when the system load is light.
2. An asterisk at the end of a line means that this disk is in a state where it is not available for new block allocation.

Parameters

Device
The device name of the file system to be queried for available file space. File system names need not be fully-qualified. `fs0` is as acceptable as `/dev/fs0`.

This must be the first parameter.

-d
List only disks that can hold data.

-F
List the number of inodes and how many of them are free.

-m
List only disks that can hold metadata.

-P *PoolName*
Lists only disks that belong to the requested storage pool.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

--block-size {*BlockSize* | auto}

BlockSize
Specifies the unit in which the number of blocks is displayed. The value must be of the form [*n*]K, [*n*]M, [*n*]G, or [*n*]T, where:

n
Is an optional variable in the range 1 - 1023.

K, M, G, T
Stand for KiB, MiB, GiB, and TiB. For example, if you specify 1M, the number of blocks is displayed in units of 1 MiB.

auto
Causes the command to automatically scale the number of blocks to an easy-to-read value.

The default value is 1K.

--qos *QosClass*
Specifies the Quality of Service for I/O operations (QoS) class to which the instance of the command is assigned. If you do not specify this parameter, the instance of the command is assigned by default to the `maintenance` QoS class. This parameter has no effect unless the QoS service is enabled. For more information, see the topic "mmchqos command" on page 263. Specify one of the following QoS classes:

maintenance
This QoS class is typically configured to have a smaller share of file system IOPS. Use this class for I/O-intensive, potentially long-running GPFS commands, so that they contribute less to reducing overall file system performance.

other
This QoS class is typically configured to have a larger share of file system IOPS. Use this class for administration commands that are not I/O-intensive.

For more information, see the topic *Setting the Quality of Service for I/O operations (QoS)* in *IBM Spectrum Scale: Administration Guide*.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

If you are a root user, the node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

If you are a non-root user, you may specify only file systems that belong to the same cluster as the node on which the `mmdf` command was issued.

Examples

1. To query all disks in the `fs2` file system that can hold data, issue this command:

```
mmdf fs2 -d
```

The system displays information similar to:

```
disk        disk size  failure holds    holds  free KB in  free KB in
name            in KB    group metadata data  full blocks  fragments
----------- --------- -------- -------- ----  -----------  ---------
Disks in storage pool: system (Maximum disk size allowed is 833 GB)
         -------------                                -------- ----------
(pool total)        0                                0 (0%)      0 (0%)

Disks in storage pool: sp1 (Maximum disk size allowed is 359 GB)
gpfs1002nsd   8897968        1 no       yes  8342016 (94%)    928 (0%)
            ---------                      ------------- ------------
(pool total)  8897968                       8342016 (94%)    928 (0%)

            =========                      ============== ============
(data)        8897968                       8342016 (94%)    928 (0%)
(metadata)          0                             0 (0%)       0 (0%)
            =========                      ============== ============
(total)       8897968                       8342016 (94%)    928 (0%)
```

2. To query all disks in the `fs1` file system with the number of blocks automatically scaled to an easy-to-read value, issue this command:

```
 mmdf fs1 --block-size auto
```

The system displays information similar to:

```
disk                disk size  failure holds    holds                 free
free
name                             group metadata data        in full blocks        in
fragments
--------------- ------------- -------- -------- ----- --------------------
-------------------
Disks in storage pool: system (Maximum disk size allowed is 437 GB)
gpfs1001nsd              33.4G        1 yes      no           33.19G ( 99%)          2.5M
( 0%)
gpfs1002nsd              33.4G        1 yes      no            33.2G ( 99%)          2.5M
( 0%)
gpfs1003nsd              33.4G        2 yes      no            33.2G ( 99%)          2.5M
( 0%)
gpfs1004nsd              33.4G        2 yes      no           33.27G (100%)          2.5M
( 0%)
gpfs1005nsd              33.4G        2 yes      no           33.32G (100%)        1.562M
( 0%)
                -------------                           --------------------
-------------------
(pool total)              167G                                166.2G (100%)        11.56M
( 0%)

Disks in storage pool: sp1 (Maximum disk size allowed is 484 GB)
gpfs1006nsd              33.4G        4 no       yes           33.4G (100%)        1.562M
( 0%)
gpfs1007nsd              33.4G        4 no       yes           33.4G (100%)        1.562M
( 0%)
gpfs1008nsd              33.4G        4 no       yes           33.4G (100%)        1.562M
( 0%)
gpfs1010nsd              33.4G        4 no       yes           33.4G (100%)        1.562M
( 0%)
gpfs1011nsd              33.4G        4 no       yes           33.4G (100%)        1.562M
( 0%)
gpfs1012nsd              33.4G        5 no       yes           33.4G (100%)        1.562M
( 0%)
gpfs1013nsd              33.4G        5 no       yes           33.4G (100%)        1.562M
( 0%)
gpfs1014nsd              33.4G        5 no       yes           33.4G (100%)        1.562M
( 0%)
gpfs1015nsd              33.4G        5 no       yes           33.4G (100%)        1.562M
( 0%)
gpfs1016nsd              33.4G        5 no       yes           33.4G (100%)        1.562M
( 0%)
                -------------                           --------------------
```

```
-------------------
(pool total)             334G                                  334G (100%)        15.62M
( 0%)

                 =============                         ====================
==================
(data)                   334G                                  334G (100%)        15.62M
( 0%)
(metadata)               167G                                 166.2G (100%)        11.56M
( 0%)
                 =============                         ====================
==================
(total)                  501G                                  500.2G (100%)        27.19M
( 0%)

Inode Information
-----------------
Number of used inodes:            4043
Number of free inodes:          497717
Number of allocated inodes:     501760
Maximum number of inodes:       514048
```

3. To query `fs1` for inode information, issue this command:

```
mmdf fs1 -F
```

The system displays information similar to:

```
Inode Information
-----------------
Number of used inodes:            4043
Number of free inodes:          497717
Number of allocated inodes:     501760
Maximum number of inodes:       514048
```

See also

- “mmchfs command” on page 232
- “mmcrfs command” on page 318
- “mmdelfs command” on page 373
- “mmlsfs command” on page 506

Location

`/usr/lpp/mmfs/bin`

mmdiag command

Displays diagnostic information about the internal GPFS state on the current node.

Synopsis

```
mmdiag [--afm [fileset={all|device[:filesetName]}|gw][-Y]]
        [--all [-Y]] [--version [-Y]] [--waiters [-Y]] [--deadlock [-Y]] [--threads [-Y]]
        [--lroc [-Y]] [--memory [-Y]] [--network [-Y]] [--config [-Y]] [--trace [-Y]]
        [--iohist [verbose] [-Y]] [--tokenmgr [-Y]] [--commands [-Y]]
        [--dmapi [session|event|token|disposition|all]]
        [--rpc [node[=name]|size|message|all|nn{S|s|M|m|H|h|D|d}] [-Y]]
        [--stats [-Y]][--nsd [all] [-Y]] [--eventproducer [-Y]]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmdiag` command to query various aspects of the GPFS internal state for troubleshooting and tuning purposes. The `mmdiag` command displays information about the state of GPFS on the node where it is executed. The command obtains the required information by querying the GPFS daemon process (`mmfsd`), and thus functions only when the GPFS daemon is running.

Results

The `mmdiag` command displays the requested information and returns 0 if successful.

Parameters

--afm
Displays status and statistics of linked AFM and AFM DR filesets that are assigned to the gateway node. Accepts the following options:

Note: If you do not specify any option, status and statistics of all filesets are displayed.

fileset=all
Displays status and statistics of all active filesets.

fileset=*device*
Displays status and statistics of all active filesets on a specified device.

fileset=*device*:*filesetName*
Displays status and statistics of a specified fileset on the specified device.

gw
Displays gateway statistics like queue length and memory.

--all
Displays all available information. This option is the same as specifying all of the `mmdiag` parameters.

--commands
Displays all the commands currently running on the local node.

--config
Displays configuration parameters and their settings. The list of configuration parameters that are shown here consists of configuration parameters that are known to `mmfsd`. Note that some configuration parameters (for example, trace settings) are scanned only by the layers of code above `mmfsd`, and those parameters are shown in `mmlsconfig` output but not here.

While `mmlsconfig` displays only a subset of configuration parameters (generally those parameters that have nondefault settings), the list here shows a larger parameter set. All of the documented

mmfsd configuration parameters are shown, plus some of the undocumented parameters (generally those parameters that are likely to be helpful in tuning and troubleshooting).

Note that parameter values that are shown here are those parameters that are currently in effect (as opposed to the values shown in mmlsconfig output, which can show the settings that become effective on the next GPFS restart).

--deadlock
Displays the longest waiters that exceed the deadlock detection thresholds.

If a deadlock situation occurs, administrators can use this information from all nodes in a cluster to help decide how to break up the deadlock.

--dmapi
Displays various DMAPI information. If no other options are specified, summary information is displayed for sessions, pending events, cached tokens, stripe groups, and events that are waiting for reply. The --dmapi parameter accepts the following options:

session
Displays a list of sessions.

event
Displays a list of pending events.

token
Displays a list of cached tokens, stripe groups, and events that waiting for reply.

disposition
Displays the DMAPI disposition for events.

all
Displays all of the session, event, token, and disposition information with more details.

Note: -Y is not supported with the **--dmapi** option.

--eventproducer
Displays statistics for file audit logging and clustered watch folder producers. The statistics include counts of how many messages have been sent, how many messages have been delivered (the target sink has acknowledged that the message has been received), messages that the producer failed to deliver, the amount of bytes sent, breakdown of the types of messages that were sent and delivered, and information on the status of the producer. For more information about the producer state and state changes, use the **mmhealth** command.

--iohist [verbose]
Displays recent I/O history. The information about I/O requests recently submitted by GPFS code is shown here. It can provide some insight into various aspects of GPFS IO, such as the type of data or metadata being read or written, the distribution of I/O sizes, and I/O completion times for individual I/Os. This information can be useful in performance tuning and troubleshooting.

verbose
Displays additional columns of information info1, info2, context, and thread. The contents of the columns are as follows:

info1, info2
The contents of columns info1 and info2 depend on the buffer type. The buffer type is displayed in the Buf type column of the command output:

Table 22. Contents of columns input1 and input2 depending on the value in column Buf type

Buf type (Buffer type)	info1	info2
data	The inode number of the file	The block number of the file
metadata	The inode number of the file	(For internal use by IBM)
LLIndBlock	The inode number of the file	(For internal use by IBM)

Table 22. Contents of columns input1 and input2 depending on the value in column Buf type (continued)

Buf type (Buffer type)	info1	info2
inode	(For internal use by IBM)	The inode number of the file
Other types, such as diskDesc, sgDesc, and others.	(For internal use by IBM)	(For internal use by IBM)

context
The I/O context that started this I/O.

thread
The name of the thread that started this I/O.

The node that the command is issued from determines the I/O completion time that is shown.

If the command is issued from a Network Shared Disk (NSD) server node, the command shows the time that is taken to complete or serve the read or write I/O operations that are sent from the client node. This refers to the latency of the operations that are completed on the disk by the NSD server.

If the command is issued on an NSD client node that does not have local access to the disk, the command shows the complete time (requested by the client node) that is taken by the read or write I/O operations to complete. This refers to the latency of I/O request to the NSD server and the latency of I/O operations that are completed on the disk by the NSD server.

If the Type of the I/O is "lrc", this indicates the I/O was made to an LROC device. Under RW column, The LR value indicates read from LROC device, while the LS indicates write to LROC device. For more information, see *Local read-only cache* in *IBM Spectrum Scale: Administration Guide*.

--lroc
Displays status and statistics for local read-only cache (LROC) devices. The statistics displayed are relevant only to the node where the command is issued from. This parameter is valid for x86_64, PPC64, and PPC64LE Linux nodes.

--memory
Displays information about mmfsd memory usage. Several distinct memory regions are allocated and used by mmfsd, and it can be important to know the memory usage situation for each one.

Heap memory that is allocated by mmfsd
This area is managed by the OS and is not associated with a preset limit that is enforced by GPFS.

Memory pools 1 and 2
Both of these pools refer to a single memory area, also known as the shared segment. It is used to cache various kinds of internal GPFS metadata and for many other internal uses. This memory area is allocated by a special, platform-specific mechanism and is shared between user space and kernel code. The preset limit on the maximum shared segment size, current usage, and some prior usage information are shown here.

Memory pool 3
This area is also known as the token manager pool. This memory area is used to store the token state on token manager servers. The preset limit on the maximum memory pool size, current usage, and some prior-usage information are shown here.

This information can be useful when you are troubleshooting ENOMEM errors that are returned by GPFS to a user application and memory allocation failures reported in a GPFS log file.

--network
Displays information about mmfsd network connections and pending Remote Procedure Calls (RPCs). Basic information and statistics about all existing mmfsd network connections to other nodes is displayed, including information about broken connections. If any RPCs are pending (that is, sent but not yet replied to), the information about each one is shown, including the list of RPC destinations and

the status of the request for each destination. This information can be helpful in following a multinode chain of dependencies during a deadlock or performance-problem troubleshooting.

--nsd [all]
Displays status and queue statistics for NSD queues that contain pending requests.

all
Displays status and queue statistics for all NSD queues.

--rpc
Displays RPC performance statistics. The `--rpc` parameter accepts the following options:

node[=*name*]
Displays all per node statistics (channel wait, send time TCP, send time verbs, receive time TCP, latency TCP, latency verbs, and latency mixed). If *name* is specified, all per node statistics for just the specified node are displayed.

size
Displays per size range statistics.

message
Displays per message type RPC execution time.

all
Displays everything.

***nn*{S|s|M|m|H|h|D|d}**
Displays per node RPC latency statistics for the latest number of intervals, which are specified by *nn*, for the interval specified by one of the following characters:

S|s
Displays second intervals only.

M|m
Displays first the second intervals since the last-minute boundary followed by minute intervals.

H|h
Displays first the second and minute intervals since their last minute and hour boundary followed by hour intervals.

D|d
Displays first the second, minute, and hour intervals since their last minute, hour, and day boundary followed by day intervals.

Averages are displayed as a number of milliseconds with three decimal places (one-microsecond granularity).

--stats
Displays some general GPFS statistics.

GPFS uses a diverse array of objects to maintain the file system state and cache various types of metadata. The statistics about some of the more important object types are shown here.

OpenFile
This object is needed to access an inode. The target maximum number of cached OpenFile objects is governed by the `maxFilesToCache` configuration parameter. Note that more OpenFile objects can be cached, depending on the workload.

CompactOpenFile
These objects contain an abbreviated form of an OpenFile, and are collectively known as *stat cache*. The target maximum number of cached CompactOpenFile objects is governed by the **maxStatCache** parameter of the **mmchconfig** command.

OpenInstance
This object is created for each open file instance (file or directory that is opened by a distinct process).

BufferDesc
This object is used to manage buffers in the GPFS page pool.

indBlockDesc
This object is used to cache indirect block data.

All of these objects use the shared segment memory. For each object type, a preset target exists, which is derived from configuration parameters and the memory available in the shared segment. The information about current object usage can be helpful in performance tuning.

--threads
Displays `mmfsd` thread statistics and the list of active threads. For each thread, its type and kernel thread ID are shown. All non-idle `mmfsd` threads are shown. For those threads that are currently waiting for an event, the wait reason and wait time in seconds are shown. This information provides more detail than the data displayed by `mmdiag --waiters`.

--tokenmgr
Displays information about token management. For each mounted GPFS file system, one or more token manager nodes is appointed. The first token manager is always colocated with the file system manager, while other token managers can be appointed from the pool of nodes with the *manager* designation. The information that is shown here includes the list of currently appointed token manager nodes and, if the current node is serving as a token manager, some statistics about prior token transactions.

--trace
Displays current trace status and trace levels. During GPFS troubleshooting, it is often necessary to use the trace subsystem to obtain the debug data necessary to understand the problem. See *Trace facility* in *IBM Spectrum Scale: Problem Determination Guide*. It is important to have trace levels set correctly, per instructions provided by the IBM Support Center. The information that is shown here makes it possible to check the state of tracing and to see the trace levels currently in effect.

--version
Displays information about the GPFS build currently running on this node. This information helps in troubleshooting installation problems. The information that is displayed here can be more comprehensive than the version information that is available from the OS package management infrastructure, in particular when an e-fix is installed.

--waiters
Displays `mmfsd` threads that are waiting for events. This information can be helpful in troubleshooting deadlocks and performance problems. For each thread, the thread name, wait time in seconds, and wait reason are typically shown. Only non-idle threads that are currently waiting for some event to occur are displayed. Note that only `mmfsd` threads are shown; any application I/O threads that might be waiting in GPFS kernel code would not be present here.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmdiag` command.

Examples

1. To display a list of waiters, enter the following command:

```
mmdiag --waiters
```

The command displays output like the following example:

```
=== mmdiag: waiters ===
0x11DA520 waiting 0.001147000 seconds, InodePrefetchWorker:
 for I/O completion
0x2AAAAAB02830 waiting 0.002152000 seconds, InodePrefetchWorker:
 for I/O completion
0x2AAAAB103990 waiting 0.000593000 seconds, InodePrefetchWorker:
 for I/O completion
0x11F51E0 waiting 0.000612000 seconds, InodePrefetchWorker:
 for I/O completion
0x11EDE60 waiting 0.005736500 seconds, InodePrefetchWorker:
 on ThMutex 0x100073ABC8 (0xFFFFC2000073ABC8)
 (CacheReplacementListMutex)
```

In this example, all waiters have a short wait duration and represent a typical snapshot of normal GPFS operation.

2. To display information about memory use, enter the following command:

```
mmdiag --memory
```

The command displays output like the following example:

```
mmfsd heap size: 1503232 bytes

current mmfsd heap bytes in use: 1919624 total 1867672 payload

Statistics for MemoryPool id 1 ("Shared Segment (EPHEMERAL)")
         128 bytes in use
   557721725 hard limit on memory usage
     1048576 bytes committed to regions
           1 allocations
           1 frees
           0 allocation failures


Statistics for MemoryPool id 2 ("Shared Segment")
     8355904 bytes in use
   557721725 hard limit on memory usage
     8785920 bytes committed to regions
     1297534 allocations
     1296595 frees
           0 allocation failures


Statistics for MemoryPool id 3 ("Token Manager")
      496184 bytes in use
   510027355 hard limit on memory usage
      524288 bytes committed to regions
        1309 allocations
         130 frees
           0 allocation failures
```

In this example, a typical memory usage picture is shown. None of the memory pools are close to being full, and no prior allocation failures occurred.

3. To display information about the network, enter the following command:

```
mmdiag --network
```

The command displays information like the following example:

```
=== mmdiag: network ===

Pending messages:
  (none)
Inter-node communication configuration:
  tscTcpPort      1191
  my address      9.114.53.217/25 (eth2) <c0n2>
```

```
  my addr list    9.114.53.217/25 (eth2)
  my node number  4
TCP Connections between nodes:
  Device null:
    hostname                      node     destination     status     err  sock  sent(MB)  recvd(MB)
ostype
    c941f1n05.pok.stglabs.ibm.com  <c0n1>   9.114.78.25     broken     233  -1    0         0
Linux/L
  Device eth2:
    hostname                      node     destination     status     err  sock  sent(MB)  recvd(MB)
ostype
    c941f3n03.pok.stglabs.ibm.com  <c0n0>   9.114.78.43     connected  0    61    0         0
Linux/L
    c870f4ap06                    <c0n3>   9.114.53.218    connected  0    64    0         0
Linux/B
Connection details:
  <c0n1> 9.114.78.25/0 (c941f1n05.pok.stglabs.ibm.com)
    connection info:
      retry(success): 0(0)
  <c0n0> 9.114.78.43/0 (c941f3n03.pok.stglabs.ibm.com)
    connection info:
      retry(success): 0(0)
      tcp connection state: established     tcp congestion state: open
    packet statistics:
      lost: 0     unacknowledged: 0
      retrans: 0     unrecovered retrans: 0
    network speed(μs):
      rtt(round trip time): 456     medium deviation of rtt: 127
    pending data statistics(byte):
      read/write calls pending: 0
      GPFS Send-Queue: 0     GPFS Recv-Queue: 0
      Socket Send-Queue: 0     Socket Recv-Queue: 0
  <c0n3> 9.114.53.218/0 (c870f4ap06)
    connection info:
      retry(success): 0(0)
      tcp connection state: established     tcp congestion state: open
    packet statistics:
      lost: 0     unacknowledged: 0
      retrans: 0     unrecovered retrans: 0
    network speed(μs):
      rtt(round trip time): 8813     medium deviation of rtt: 13754
    pending data statistics(byte):
      read/write calls pending: 0
      GPFS Send-Queue: 0     GPFS Recv-Queue: 0
      Socket Send-Queue: 0     Socket Recv-Queue: 0
Device details:
  devicename     speed     mtu       duplex    rx_dropped rx_errors tx_dropped tx_errors
  eth2           1000      1500      full      0          0         0          0
diag verbs: VERBS RDMA class not initialized
```

4. To display information about status and statistics of all AFM and AFM DR relationships, enter the following command: **mmdiag --afm** The command displays output similar to the following example:

```
=== mmdiag: afm ===
 AFM Gateway: p7fbn10 Active

 AFM-Cache:  adrFset-4 (/gpfs/fs1/adrFset-4) in Device: fs1
    Mode: primary
    Home: p7fbn09 (nfs://p7fbn09/gpfs/fs1/adrFset-4)
    Fileset Status: Linked
         Handler-state: Mounted
         Cache-state: PrimInitInProg
         Q-state: Normal Q-length: 12126378 Q-executed: 40570
 AFM-Cache:  adrFset-5 (/gpfs/fs1/adrFset-5) in Device: fs1
    Mode: primary
    Home: p7fbn09 (nfs://p7fbn09/gpfs/fs1/adrFset-5)
    Fileset Status: Linked
         Handler-state: Mounted
         Cache-state: PrimInitInProg
         Q-state: Normal Q-length: 6164585 Q-executed: 7113648
 AFM-Cache:  adrFset-10 (/gpfs/fs1/adrFset-10) in Device: fs1
    Mode: primary
    Home: p7fbn09 (nfs://p7fbn09/gpfs/fs1/adrFset-10)
    Fileset Status: Linked
         Handler-state: Mounted
         Cache-state: PrimInitInProg
         Q-state: Normal Q-length: 16239687 Q-executed: 2415474
```

5. To display gateway statistics, enter the following command: **mmdiag --afm gw** The command displays output similar to the following example:

```
=== mmdiag: afm ===
 AFM Gateway: p7fbn10 Active
```

```
QLen: 33165776 QMem: 12560682162 SoftQMem: 12884901888 HardQMem 32212254720
Ping thread: Started
```

6. To display LROC statistics, enter the following command: **mmdiag --lroc** The command displays output similar to the following example:

```
=== mmdiag: lroc ===
LROC Device(s): '090BD5CD603456D2#/dev/nvme1n1;090BD5CD60354659#/dev/nvme0n1;' status Running
Cache inodes 1 dirs 1 data 1  Config: maxFile -1 stubFile -1
Max capacity: 3051313 MB, currently in use: 3559 MB
Statistics starting from: Tue Feb 23 11:42:32 2021

      Inode objects stored 312454 (1220 MB) recalled 157366 (614 MB) = 50.36 %
      Inode objects queried 0 (0 MB) = 0.00 % invalidated 157460 (615 MB)
      Inode objects failed to store 6 failed to recall 0 failed to query 0 failed to inval 0

      Directory objects stored 84 (2 MB) recalled 979 (226 MB) = 1165.48 %
      Directory objects queried 0 (0 MB) = 0.00 % invalidated 80 (6 MB)
      Directory objects failed to store 0 failed to recall 0 failed to query 0 failed to
inval 0 inval no recall 0

      Data objects stored 57412 (188807 MB) recalled 10150641 (40597918 MB) = 17680.35 %
      Data objects queried 1 (0 MB) = 100.00 % invalidated 57612 (228070 MB)
      Data objects failed to store 30 failed to recall 407 failed to query 0 failed to inval
0 inval no recall 54307

  agent inserts=1074934, reads=162501285
        response times (usec):
        insert min/max/avg=3/8030/121
        read   min/max/avg=1/66717/2919

  ssd   writeIOs=380060, writePages=48671744
        readIOs=10345039194, readPages=10124700092
        response times (usec):
        write  min/max/avg=192/11985/233
        read   min/max/avg=13/49152/225
```

Location

/usr/lpp/mmfs/bin

mmdsh command

Runs commands on multiple nodes or network connected hosts at the same time.

Synopsis

```
mmdsh -N {Node[,Node...] | NodeFile | NodeClass}
      [-l LoginName] [-i] [-s] [-r RemoteShellPath]
      [-v [-R ReportFile]] [-f FanOutValue] Command
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmdsh` command to remotely execute a command concurrently on each of the nodes that are specified in the `-N` option.

Note: For security reasons, the `mmdsh` command is limited to the list of allowable remote commands when sudo wrappers are implemented. For more information on how to configure sudo wrappers, see *Configuring sudo* in *IBM Spectrum Scale: Administration Guide*.

CAUTION: The mmdsh command runs any authorized command that you specify concurrently against the list of nodes that you specify. To avoid accidentally damaging or corrupting your clusters or file systems, ensure that you have specified the correct command and the correct list of nodes before you run mmdsh.

Parameters

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}

Runs the command on the nodes in the given node specification. The `nodespecification` argument can be a comma-separated list of nodes, a node file, or a node class. The nodes in the list or the file can be specified as long or short admin or daemon node names, node numbers, node number ranges, or IP addresses.

-l *LoginName*

Allows the user to specify a log-in name for the nodes. The log-in names are entered in the command line.

-i

Displays the set of nodes before the command is run on those nodes.

-s

Suppresses the prepending of the hostname string to each line of output generated by running the command on the remote node.

-r *RemoteShellPath*

Specifies the full path of the remote shell command that must be used.

-v

Verifies that a node is reachable with an ICMP echo command (network ping) before adding it to the set of nodes on which the command must be run.

-R *ReportFile*

Reports the list of hosts removed from the working collective when host verification (host ping) fails. The report is written to the specified file with one host per line. The report is generated only when combined with the **-v** parameter.

-f *FanOutValue*

Specifies a fanout value used for concurrent execution. The default value is 64.

Command

To be run on the remote hosts.

Exit status

The return code from this command does not reliably indicate the success or failure of the command that is executed on the remote node. To determine the overall command status, review the messages that are returned by the remote commands.

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmdsh` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Example

1. To run the command on a list of nodes specified in NodeFile, run the following command:

```
mmdsh -N ./nodes uname -a
```

2. To run the command on a specified list of hosts, run the following command:

```
mmdsh -N host1,host2,host3 ls
```

3. To run the command on the quorum nodes specified by a node class, run the following command:

```
mmdsh -N quorumnodes date
```

4. To run the command on the hosts listed in the hostfile, run the following command:

```
mmdsh -N ./nodes -r /usr/bin/ssh uname -a
```

5. To run the command on a specified node along with a log-in name, run the following command:

```
mmdsh -N host -l admin ls
```

mmeditacl command

Creates or changes a GPFS access control list.

Synopsis

```
mmeditacl [-d] [-k {nfs4 | posix | native}] Filename
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmeditacl` command for interactive editing of the ACL of a file or directory. This command uses the default editor, specified in the EDITOR environment variable, to display the current access control information, and allows the file owner to change it. The command verifies the change request with the user before making permanent changes.

This command cannot be run from a Windows node.

The EDITOR environment variable must contain a complete path name, for example:

```
export EDITOR=/bin/vi
```

For information about NFS V4 ACLs, see the topics *Managing GPFS access control lists* and *NFS and GPFS* in the *IBM Spectrum Scale: Administration Guide*.

Users may need to see ACLs in their true form as well as how they are translated for access evaluations. There are four cases:

1. By default, `mmeditacl` returns the ACL in a format consistent with the file system setting, specified using the `-k` flag on the `mmcrfs` or `mmchfs` commands.
   - If the setting is `posix`, the ACL is shown as a traditional ACL.
   - If the setting is `nfs4`, the ACL is shown as an NFS V4 ACL.
   - If the setting is `all`, the ACL is returned in its true form.
2. The command `mmeditacl -k nfs4` always produces an NFS V4 ACL.
3. The command `mmeditacl -k posix` always produces a traditional ACL.
4. The command `mmeditacl -k native` always shows the ACL in its true form regardless of the file system setting.

The following describes how `mmeditacl` works for POSIX and NFS V4 ACLs:

```
Command               ACL    mmcrfs -k  Display        -d (default)
-------------------   -----  ---------  -------------  --------------
mmeditacl             posix  posix      Access ACL     Default ACL
mmeditacl             posix  nfs4       NFS V4 ACL     Error[1]
mmeditacl             posix  all        Access ACL     Default ACL
mmeditacl             nfs4   posix      Access ACL[2]  Default ACL[2]
mmeditacl             nfs4   nfs4       NFS V4 ACL     Error[1]
mmeditacl             nfs4   all        NFS V4 ACL     Error[1]
mmeditacl -k native   posix  any        Access ACL     Default ACL
mmeditacl -k native   nfs4   any        NFS V4 ACL     Error[1]
mmeditacl -k posix    posix  any        Access ACL     Default ACL
mmeditacl -k posix    nfs4   any        Access ACL[2]  Default ACL[2]
mmeditacl -k nfs4     any    any        NFS V4 ACL     Error[1]
------------------------------------------------------------------------
[1] NFS V4 ACLs include inherited entries. Consequently, there cannot
    be a separate default ACL.
[2] Only the mode entries (owner, group, everyone) are translated.
    The rwx values are derived from the
    NFS V4 file mode attribute. Since the NFS V4 ACL is more granular
```

```
    in nature, some information is lost in this translation.
------------------------------------------------------------------
```

In the case of NFS V4 ACLs, there is no concept of a default ACL. Instead, there is a single ACL and the individual access control entries can be flagged as being inherited (either by files, directories, both, or neither). Consequently, specifying the `-d` flag for an NFS V4 ACL is an error. By its nature, storing an NFS V4 ACL implies changing the inheritable entries (the GPFS default ACL) as well.

Depending on the file system's `-k` setting (`posix`, `nfs4`, or `all`), `mmeditacl` may be restricted. The `mmeditacl` command is not allowed to store an NFS V4 ACL if `-k posix` is in effect, and is not allowed to store a POSIX ACL if `-k nfs4` is in effect. For more information, see the description of the `-k` flag for the `mmchfs`, `mmcrfs`, and `mmlsfs` commands.

Parameters

Filename
The path name of the file or directory for which the ACL is to be edited. If the `-d` option is specified, *Filename* must contain the name of a directory.

Options

-d
Specifies that the default ACL of a directory is to be edited.

-k {nfs4 | posix | native}

nfs4
Always produces an NFS V4 ACL.

posix
Always produces a traditional ACL.

native
Always shows the ACL in its true form regardless of the file system setting.

This option should not be used for routine ACL manipulation. It is intended to provide a way to show the translations that are done. For example, if a `posix` ACL is translated by NFS V4. Beware that if the `-k nfs4` flag is used, but the file system does not allow NFS V4 ACLs, you will not be able to store the ACL that is returned. If the file system does support NFS V4 ACLs, the `-k nfs4` flag is an easy way to convert an existing `posix` ACL to `nfs4` format.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You may issue the `mmeditacl` command only from a node in the GPFS cluster where the file system is mounted.

The `mmeditacl` command may be used to display an ACL. POSIX ACLs may be displayed by any user with access to the file or directory. NFS V4 ACLs have a `READ_ACL` permission that is required for non-privileged users to be able to see an ACL. To change an existing ACL, the user must either be the owner, the root user, or someone with control permission (`WRITE_ACL` is required where the existing ACL is of type NFS V4).

Examples

To edit the ACL for a file named `project2.history`, issue this command:

```
mmeditacl project2.history
```

The current ACL entries are displayed using the default editor, provided that the EDITOR environment variable specifies a complete path name. When the file is saved, the system displays information similar to:

```
mmeditacl: 6027-967 Should the modified ACL be applied? (yes) or (no)
```

After responding `yes`, the ACLs are applied.

See also

- “mmdelacl command” on page 360
- “mmgetacl command” on page 428
- “mmputacl command” on page 617

Location

`/usr/lpp/mmfs/bin`

mmedquota command

Sets quota limits.

Synopsis

```
mmedquota {-u [-p [ProtoFileset:]ProtoUser] [Device:Fileset:]User ... |
           -g [-p [ProtoFileset:]ProtoGroup] [Device:Fileset:]Group ... |
           -j [-p ProtoFileset] Device:Fileset ... |
           -d {-u User ... | -g Group ... | -j Device:Fileset ...} |
           -t {{-u | -g | -j} [--reset]}}
```

Availability

Available on all IBM Spectrum Scale editions. Available on AIX and Linux.

Description

The `mmedquota` command serves two purposes:

1. Sets or changes quota limits or grace periods for users, groups, and filesets in the cluster from which the command is issued.
2. Reestablishes user, group, or fileset default quotas for all file systems with default quotas enabled in the cluster.

Important: Quota limits are not enforced for root users (by default). For information on managing quotas, see *Managing GPFS quotas* in the *IBM Spectrum Scale: Administration Guide*.

The `mmedquota` command displays the current values for these limits, if any, and prompts you to enter new values with the default editor:

- The current block usage: The amount of disk space that is used by this user, group, or fileset, in 1 KB units; display only.
- The current inode usage: Display only.
- Node soft limit.
- Inode hard limit.
- Block soft limit: The amount of disk space that this user, group, or fileset is allowed to use during normal operation.
- Block hard limit: The amount of disk space that this user, group, or fileset is allowed to use during the grace period.

  Note on block limits:

  – The command displays the current block limits in KB.

  – When you specify a block limit, you can add a suffix to the number to indicate the unit of measure: `g`, `G`, `k`, `K`, `m`, `M`, `p`, `P`, `t`, or `T`. If you do not specify a suffix, the command assumes that the number is in bytes.

  – The maximum block limit is 999999999999999 K (about 931322 T). For values greater than 976031318016 K (909 T) you must specify the equivalent value with the suffix K, M, or G or without any suffix.

  Note: A block or inode limit of 0 indicates no limit.

The `mmedquota` command waits for the edit window to be closed before you check and apply new values. If an incorrect entry is made, reissue the command and enter the correct values.

You can also use the `mmedquota` command to change the file system-specific grace periods for block and file usage if the default of one week is unsatisfactory. The grace period is the time during which users can

exceed the soft limit. If the user, group, or fileset does not show reduced usage below the soft limit before the grace period expires, the soft limit becomes the new hard limit.

When you set quota limits for a file system, consider replication in the file system. See the topic *Listing quotas* in the *IBM Spectrum Scale: Administration Guide*.

The EDITOR environment variable must contain a complete path name, for example:

```
export EDITOR=/bin/vi
```

Parameters

Device
Specifies the device name of the file system for which quota information is to be displayed. File system names need not be fully qualified. `fs0` is as acceptable as `/dev/fs0`.

Fileset
Specifies the name of a fileset that is on the *Device* for which quota information is to be displayed.

User
Name or user ID of target user for quota editing.

Group
Name or group ID of target group for quota editing.

-d
Reestablish default quota limits for a specific user, group, or fileset that had an explicit quota limit set by a previous invocation of the `mmedquota` command.

-g
Sets quota limits or grace times for groups.

-j
Sets quota limits or grace times for filesets.

-p
Applies already-established limits to a particular user, group, or fileset.

When invoked with the `-u` option, [*ProtoFileset*:]*ProtoUser* limits are automatically applied to the specified *User* or space-delimited list of users.

When invoked with the `-g` option, [*ProtoFileset*:]*ProtoGroup* limits are automatically applied to the specified *Group* or space-delimited list of groups.

When invoked with the `-j` option, *ProtoFileset* limits are automatically applied to the specified fileset or space-delimited list of fileset names.

You can specify any user as a *ProtoUser* for another *User*, or any group as a *ProtoGroup* for another *Group*, or any fileset as a *ProtoFileset* for another *Fileset*.

`-p` cannot propagate a prototype quota from a user, group, or fileset on one file system to a user, group, or fileset on another file system.

-t
Sets grace period during which quotas can exceed the soft limit before it is imposed as a hard limit. The default grace period is one week.

This flag is followed by one of the following flags: `-u`, `-g`, or `-j` to specify whether the changes apply to users, groups, or filesets.

-u
Sets quota limits or grace times for users.

--reset
With this option, when grace time is modified, all relative quota entries are scanned and updated if necessary; without this option, when grace time is updated, quota entries are not scanned and updated.

Note:

- The maximum files limit is 2147483647.
- See the *Note on block limits* earlier in this topic.
- If you want to display the current grace period, issue the command `mmrepquota -t`.

Exit status

0
Successful completion.

Nonzero
A failure has occurred.

Security

You must have root authority to run the `mmedquota` command.

GPFS must be running on the node from which the `mmedquota` command is issued.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To set user quotas for user ID `pfs001`, issue this command:

```
mmedquota -u pfs001
```

   The system displays the following information:

```
*** Edit quota limits for USR pfs001
NOTE: block limits will be rounded up to the next multiple of the block size.
      block units may be: K, M, G, T or P, inode units may be: K, M or G.
gpfs3: (root): blocks in use: 8K, limits (soft = 0K, hard = 0K)
        inodes in use: 1, limits (soft = 0, hard = 0)
gpfs2: (fset4): blocks in use: 4104K, limits (soft = 102400K, hard = 153600K)
        inodes in use: 2, limits (soft = 100, hard = 150)
gpfs2: (fset3): blocks in use: 0K, limits (soft = 0K, hard = 0K)
        inodes in use: 0, limits (soft = 0, hard = 0)
gpfs2: (root): blocks in use: 0K, limits (soft = 0K, hard = 0K)
        inodes in use: 0, limits (soft = 0, hard = 0)
gpfs1: (fset1): blocks in use: 0K, limits (soft = 256K, hard = 256K)
        inodes in use: 0, limits (soft = 30, hard = 40)
gpfs1: (root): blocks in use: 0K, limits (soft = 256K, hard = 256K)
        inodes in use: 0, limits (soft = 40, hard = 45)
```

2. To reset default group quota values for the group `blueteam`, issue this command:

```
mmedquota -d -g blueteam
```

   To verify the change, issue this command:

```
mmrepquota -q fs1
```

   The system displays the following information:

```
fs1: USR quota is on; default quota is on
fs1: GRP quota is on; default quota is on
fs1: FILESET quota is on; default quota is off
```

3. To change the grace periods for all users, issue this command:

```
mmedquota -t -u
```

The system displays the following information in your default editor:

```
*** Edit grace times:
Time units may be : days, hours, minutes, or seconds
Grace period before enforcing soft limits for USRs:
gpfs0: block grace period: 7 days, file grace period: 7 days
```

4. To set user quotas for device gpfs2, fileset fset3, and userid pfs001, issue this command:

```
mmedquota -u gpfs2:fset3:pfs001
```

The system displays the following information:

```
*** Edit quota limits for USR gpfs2:fset3:pfs001
NOTE: block limits will be rounded up to the next multiple of the block size.
      block units may be: K, M, G, T or P, inode units may be: K, M or G.
gpfs2: (fset3): blocks in use: 0K, limits (soft = 0K, hard = 0K)
        inodes in use: 0, limits (soft = 0, hard = 0)
```

5. To apply already-established limits of user pfs002 to user pfs001, issue this command:

```
mmedquota -u -p pfs002 pfs001
```

The system displays the following information:

```
Already established limits of protouser pfs002 in root fileset are applied
to all filesets (including root fileset) in all corresponding per-fileset
quota enabled filesystems.
```

6. To apply already-established limits of user pfs002 in fileset fset2 to user pfs001 in fileset fset1 and file system fs1, issue this command:

```
mmedquota -u -p fset2:pfs002 fs1:fset1:pfs001
```

The system displays the following information:

```
Limits of protouser pfs002 in fileset fset2 in filesystem fs1 are applied
to user pfs001 in fileset fset1
```

7. To apply an already-established fileset quota (from fileset1 to fileset2) in two different file systems (gpfstest1 and gpfstest2), issue this command:

```
mmedquota -j -p fileset1 gpfstest1:fileset2 gpfstest2:fileset2
```

See also

- “mmcheckquota command” on page 220
- “mmdefedquota command” on page 345
- “mmdefquotaoff command” on page 349
- “mmdefquotaon command” on page 352
- “mmlsquota command” on page 535
- “mmquotaon command” on page 654
- “mmquotaoff command” on page 651
- “mmrepquota command” on page 666

Location

/usr/lpp/mmfs/bin

mmexportfs command

Retrieves the information needed to move a file system to a different cluster.

Synopsis

```
mmexportfs {Device | all} -o ExportfsFile
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmexportfs` command, in conjunction with the `mmimportfs` command, can be used to move one or more GPFS file systems from one GPFS cluster to another GPFS cluster, or to temporarily remove file systems from the cluster and restore them at a later time. The `mmexportfs` command retrieves all relevant file system and disk information and stores it in the file specified with the `-o` parameter. This file must later be provided as input to the `mmimportfs` command. When running the `mmexportfs` command, the file system must be unmounted on all nodes.

When `all` is specified in place of a file system name, any disks that are not associated with a file system will be exported as well.

Exported file systems remain unusable until they are imported back with the `mmimportfs` command to the same or a different GPFS cluster.

Results

Upon successful completion of the `mmexportfs` command, all configuration information pertaining to the exported file system and its disks is removed from the configuration data of the current GPFS cluster and is stored in the user specified file *ExportfsFile*.

Parameters

***Device* | all**
: The device name of the file system to be exported. File system names need not be fully-qualified. `fs0` is as acceptable as `/dev/fs0`. Specify `all` to export all GPFS file systems, as well as all disks that do not currently belong to a file system.

  If the specified file system device is a IBM Spectrum Scale RAID-based file system, then all affected IBM Spectrum Scale RAID objects will be exported as well. This includes recovery groups, declustered arrays, vdisks, and any other file systems that are based on these objects. For more information about IBM Spectrum Scale RAID, see *IBM Spectrum Scale RAID: Administration*.

  This must be the first parameter.

-o *ExportfsFile*
: The path name of a file to which the file system information is to be written. This file must be provided as input to the subsequent `mmimportfs` command.

Exit status

0
: Successful completion.

nonzero
: A failure has occurred.

Security

You must have root authority to run the `mmexportfs` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

To export all file systems in the current cluster, issue this command:

```
mmexportfs all -o /u/admin/exportfile
```

The output is similar to this:

```
mmexportfs: Processing file system fs1 ...

mmexportfs: Processing file system fs2 ...

mmexportfs: Processing disks that do not belong to any file system ...
mmexportfs: 6027-1371 Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

See also

- “mmimportfs command” on page 465

Location

`/usr/lpp/mmfs/bin`

mmfsck command

Checks and repairs a GPFS file system.

Synopsis

```
mmfsck Device [-n | -y | --patch] [--patch-file FileName]
       [-c | -m | --skip-inode-check | --skip-directory-check]
       [-s | -v | -V] [-t TmpDirPath] [--threads Num] [--estimate-only]
       [--use-stale-replica] [--qos QosClass]
       [-N {all | mount | Node[,Node...] | NodeFile | NodeClass}]
```

or

```
mmfsck Device -n -o
       [-s | -v | -V] [-t TmpDirPath] [--threads Num]
       [-N {all | mount | Node[,Node...] | NodeFile | NodeClass}]
```

or

```
mmfsck Device --status-report
```

The file system must be unmounted before you can run the `mmfsck` command with any option other than `-o`.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmfsck` command in offline mode is intended to be used only in situations where disk or communications failures cause `MMFS_FSSTRUCT` error log entries to be issued, or where you know that disks have been forcibly removed or are otherwise permanently unavailable for use in the file system and you see other unexpected symptoms. In general, it is unnecessary to run `mmfsck` in offline mode except under the direction of the IBM Support Center. For more information about error logs, see the topic *Operating system error logs* in the *IBM Spectrum Scale: Problem Determination Guide*.

Note: When the `mmhealth` command displays an `fsstruct` error, the command prompts you to run a file system check. When the problem is resolved, issue the following command to clear the `fsstruct` error from the `mmhealth` command. You must specify the file system name twice:

```
mmsysmonc event filesystem fsstruct_fixed <filesystem_name> <filesystem_name>
```

If neither the `-n` nor `-y` flag is specified, the `mmfsck` command runs interactively and prompts you for permission to repair each consistency error as it is reported. It is suggested that you run the `mmfsck` command interactively (the default) in all but the most severely damaged file systems.

I/O errors or error messages with an instruction to run the `mmfsck` command might indicate file system inconsistencies. If so, issue the `mmfsck` command to check file system consistency and to interactively repair the file system.

For information about file system maintenance and repair, see the topic *Checking and repairing a file system* in the *IBM Spectrum Scale: Administration Guide*. In online mode the `mmfsck` command checks for the following inconsistencies:

- Blocks that are marked as allocated but that do not belong to any file (lost blocks). The corrective action is to mark the block free in the allocation map. A possible indicator of lost blocks is that I/O operations fail with an out-of-space error after repeated node failures.

- Corruptions in the block allocation map. The corrective action is to repair the corrupted blocks. This check detects the structural corruptions, such as corrupt block headers and corrupt disk addresses. It does not detect the non-structural corruptions, such as bad allocation map bits. Possible indicators of corruptions in the block allocation map are MMFS_FSSTRUCT entries in the system logs after an attempt to create or delete a file.

In offline mode the command checks for the two inconsistencies in the previous list and also checks for the following inconsistencies:

- Files for which an inode is allocated and no directory entry exists (orphaned files). The corrective action is to create directory entries for these files in a `lost+found` subdirectory of the fileset to which the orphaned file or directory belongs. The index number of the inode is assigned as the name. If you do not allow the `mmfsck` command to reattach an orphaned file, it asks for permission to delete the file.
- Directory entries that point to an inode that is not allocated. The corrective action is to remove the directory entry.
- Incorrectly formed directory entries. A directory file contains the inode number and the generation number of the file to which it refers. When the generation number in the directory does not match the generation number that is stored in the inode of the file, the corrective action is to remove the directory entry.
- Incorrect link counts on files and directories. The corrective action is to update them with accurate counts.
- Policy files that are not valid. The corrective action is to delete the file.
- Various problems that are related to filesets. Such problems include missing or corrupted fileset metadata, inconsistencies in directory structure related to filesets, missing or corrupted fileset root directories, or other problems in internal data structures. The repaired filesets are renamed as `Fileset` *FilesetId* and put into unlinked state.

You can also run the **mmfsck** command in read-only offline mode to get a list of inodes within filesets that might be deleted or modified by the repair. You can mount the file system in read-only mode and run the **tsfindinode** command to find the path names (which allow you to restore a file from backup after repair) of all affected inodes. Then run **mmfsck** again in repair mode to fix any problems with the file system. If you do not need to know the path names of affected files you can use this command for a full scan to save time: `mmfsck repair (-y)`

Use the `--patch` repair option to expedite a **mmfsck** repair by using data for corrupt inodes that has been detected during a previous run of read-only offline **mmfsck** instead of scanning the file system again for corruptions. Generally, the `--patch` option is much faster than a full **mmfsck** scan repair. However:

1. The `--patch` option is faster only for large file systems. For smaller file systems you can use either the `--patch` option or the `-y` option.
2. The `--patch` option uses a single thread to repair the file system. This can be slower than the full scan repair of the file system if there are many (more than 10,000) corruptions.
3. The `--patch` option is safe only if the file system has not been modified after generating the `--patch-file` from a previous read-only offline **mmfsck** run. Because the `--patch` option uses a list of known corruptions, if the file system is changed (for example, by mounting it as read-write after the read-only **mmfsck** run), it does not know about any new corruptions that are introduced because of existing inconsistencies in the file system. This means that it can only perform a partial repair.

   Important: This might give the impression that file system is clean even if it is not.
4. The `--patch` option cannot be used to repair certain types of corruptions in system files. In some cases, you might have to run a full scan repair. This can be determined by viewing the last record of the `--patch-file` that contains the string `need_full_fsck_scan`:
   - If it is `false`, the `--patch` option can be used.
   - If it is `true`, the `-y` option must be used.

If you do plan to run read-only offline **mmfsck** and are not sure which approach to consider, use the `--patch-file` option. Once the patch file is generated, you can use it to repair the file system.

If you are repairing a file system because of node failure and if quotas are enabled in the file system, it is a good idea to run the mmcheckquota command to make sure that the quota accounting is consistent.

Indications that might lead you to run the mmfsck command include the following events:

- An MMFS_FSSTRUCT along with an MMFS_SYSTEM_UNMOUNT error log entry on any node that indicates that some critical piece of the file system is inconsistent. For more information about error logs, see the topic *Operating system error logs* in the *IBM Spectrum Scale: Problem Determination Guide*.
- Disk media failures.
- Partial disk failure.
- EVALIDATE=214, which indicates that an invalid checksum or other consistency check failure either occurred on a disk data structure, or was reported in an error log, or was returned to an application.

For more information on recovery actions and how to contact the IBM Support Center, see the *IBM Spectrum Scale: Problem Determination Guide*.

While the mmfsck command is working, you can run another instance of mmfsck with the --status-report parameter at any time to display a consolidated status report from all the nodes that are participating in the mmfsck run. For more information, see the --status-report parameter.

Results

If the file system is inconsistent, the mmfsck command displays information about the inconsistencies and, depending on the option that is entered, might prompt you for permission to repair them. The mmfsck command tries to avoid actions that might result in loss of data. However, in some cases it might report the destruction of a damaged file.

All corrective actions require that the file system be unmounted on all nodes. If the mmfsck command is run on a mounted file system, inconsistencies are only reported, not repaired.

If a bad disk is detected, the mmfsck command stops the disk and writes an entry to the error log. The operator must manually start and resume the disk when the problem is fixed.

The file system must be unmounted on all nodes before the mmfsck command can repair file system inconsistencies.

When the command is running in verbose or semi-verbose mode, the command provides a summary of the errors and the severity of each error:

CRITICAL
The command found a critical corruption in the file system. Using the file system without repairing the corruption might cause more corruptions or file system panics.

NONCRITICAL
The command found a non-critical corruption in the file system. You can use the file system without repairing the corruption, but some metadata might not be accessible.

HARMLESS
The command found a harmless problem in the file system that indicates that some unused metadata can be freed to reclaim space.

Parameters

Device
The device name of the file system to be checked and repaired. File system names need not be fully qualified. fs0 is as acceptable as /dev/fs0.

This parameter must be the first parameter.

-n
Specifies a no response to all file system error repair prompts from the mmfsck command. This option reports inconsistencies but it does not change the file system. To save this information, redirect it to an output file when you issue the mmfsck command.

Note: If the `mmfsck` command is run offline with the **-n** parameter and it detects errors, it panics the file system to force the file system to go through cleanup before any new command can be started.

-y
Specifies a `yes` response to all file system error repair prompts from the `mmfsck` command. Use this option only on severely damaged file systems. It allows the `mmfsck` command to take any action that is necessary for repairs. This option can only be used when running the **mmfsck** command in the offline mode.

--patch
Specifies that the file system will be repaired using the information stored in the patch file that is specified with `--patch-file` *FileName*.

Note: Use of the `--patch` option causes the command to run only on the node where the command is executed as a single instance. Any information provided with the `-N` option is ignored.

--patch-file *FileName*
Specifies the name of a patch file. When the `--patch` parameter is not specified, information about file system inconsistencies (detected during an `mmfsck` run with the `-n` parameter) is stored in the patch file that is specified by *Path*. *Path* must be accessible from the file system manager node. The information that is stored in the patch file can be viewed as a report of the problems in the file system. For more information about patch files, see the topic *Checking and repairing a file system* in the *IBM Spectrum Scale: Administration Guide*.

When this option is specified with the `--patch` parameter, the information in the patch file is read and used to repair the file system.

-c
If the file system log is lost and the file system is replicated, this option causes the `mmfsck` command to attempt corrective action by comparing the replicas of metadata and data. If this error condition occurs, it is indicated by an error log entry.

-m
Has the same meaning as `-c`, except that `mmfsck` checks only the metadata replica blocks. Therefore it runs faster than with `-c`.

-o
Runs the command in online mode. See the list of conditions that the command can detect in online mode in the `Description` section earlier in this topic.

Important: If this option reports any inconsistencies, contact IBM Support for guidance.

Note: The repair functionality of the **mmfsck** command in online mode has been deprecated and it is no longer available.

--skip-inode-check
Causes the command to run faster by skipping its inode-check phase. Include this option only if you know that the inodes are valid and that only directories need to be checked. In this mode, the product does not scan all parts of the file system and therefore might not detect all corruptions in the file system.

--skip-directory-check
Causes the command to run faster by skipping its directory-check phase. Include this option if you want to check only the inodes. In this mode, the product does not scan all parts of the file system and therefore might not detect all corruptions in the file system.

-s
Specifies that the output is semi-verbose.

-v
Specifies that the output is verbose.

-V
Specifies that the output is verbose and contains information for debugging purposes.

-t *TmpDirPath*
Specifies the directory that GPFS uses for temporary storage during `mmfsck` command processing. This directory must be available on all nodes that are participating in `mmfsck` and that are designated as either a manager or quorum node. In addition to the location requirement, the storage directory has a minimum space requirement of 4 GB. The default directory for `mmfsck` processing is `/tmp`.

--threads *Num*
The number of threads that are created to run `mmfsck`. The default is 16.

--estimate-only
Causes the command to display an estimate of the time that would be required to run the `mmfsck` command in offline mode with the parameters that you are specifying. To use this option, type the `mmfsck` command on the command line with the parameters that you plan to use; include the `--estimate-only` option; and press Enter. The command displays the time estimate and does not do a file system scan.

This feature is available even while the target file system is mounted. The estimate is based on the parameters that you specify for the `mmfsck` command, on the characteristics of the target file system, and on the disk and network I/O throughput of the nodes that would participate in running the `mmfsck` command.

Note: To enable this feature, you must update all the nodes in the cluster to IBM Spectrum Scale 5.0.2 or later.

--use-stale-replica
Specifies whether to read possible stale replica blocks from unrecovered disks. The default behavior is for offline `fsck` to not read replica blocks from disks that are in unrecovered state and with a lower failure-config-version than other blocks in the same replica set. This is because disks with a lower failure-config-version are likely to have stale versions of a block. In case the higher failure-config-version disk is damaged, the block read fails and `fsck` reports it as a data loss. In such situations, using this option overrides default offline `fsck` behavior and allows `fsck` to read possible stale replica blocks from unrecovered disks. This is useful if the data loss when using this option is found to be less than the default behavior. This can be determined by running read-only offline `fsck` twice, once without `--use-stale-replica` and once with `--use-stale-replica`. Then compare the number of corruptions reported in each case.

--qos *QosClass*
Specifies the Quality of Service for I/O operations (QoS) class to which the instance of the command is assigned. If you do not specify this parameter, the instance of the command is assigned by default to the `maintenance` QoS class. This parameter has no effect unless the QoS service is enabled. For more information, see the topic "mmchqos command" on page 263. Specify one of the following QoS classes:

`maintenance`
This QoS class is typically configured to have a smaller share of file system IOPS. Use this class for I/O-intensive, potentially long-running GPFS commands, so that they contribute less to reducing overall file system performance.

other
This QoS class is typically configured to have a larger share of file system IOPS. Use this class for administration commands that are not I/O-intensive.

For more information, see the topic *Setting the Quality of Service for I/O operations (QoS)* in *IBM Spectrum Scale: Administration Guide*.

[-N {`all` | `mount` | *Node*[,*Node*...] | *NodeFile* | *NodeClass*}]
Specifies the nodes that are to participate in the check and repair of the file system. This command supports all defined node classes. The default is `all` or the current value of the `defaultHelperNodes` parameter of the `mmchconfig` command.

For information on how to specify node names, see the topic *Specifying nodes as inputs to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

--status-report
Displays a consolidated status report with information from all the nodes that are participating in the scan of the file system. While a long-running instance of **mmfsck** is running, you can run **mmfsck** with the **--status-report** parameter to verify that the long-running instance is still working and to get the status. You can issue this command from any node in the cluster, even if the node is not participating in the **mmfsck** run.

Note: To enable this feature, you must update all the nodes in the cluster to IBM Spectrum Scale 5.0.0 or later.

Exit status

0
Successful completion.

2
The command was interrupted before it completed checks or repairs.

4
The command changed the file system and it must now be restarted.

8
The file system contains damage that was not repaired.

16
The problem cannot be fixed.

64
Do a full offline file system check to verify the integrity of the file system.

The exit string is a combination of three error values:

1. The value of the Exit `errno` variable.
2. An internal value that helps to explain the source of the value in the `errno` variable.
3. The OR of several status bits.

Security

You must have root authority to run the `mmfsck` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see the topic *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

Examples

1. The following command checks file system `fs2` and displays inconsistencies but does not try to make repairs:

```
mmfsck fs2 -vn
```

The command displays output similar to the following example:

```
Checking "fs2"
FsckFlags 0x2000009
Stripe group manager <c0n0>
NeedNewLogs 0

...

Error in inode 38662 snap 0: has empty EA overflow block
Delete EA overflow block? No

...

Lost blocks were found.
```

```
Correct the allocation map? No
Block 1:332288 has map status 0x9FFFFF01 should be 0x9FFFFFFF
Block 3:332288 has map status 0x9FFFFF01 should be 0x9FFFFFFF
Checking inode map file

Corrections are needed in the inode allocation map.
Correct the allocation map? No
Inode 131075 is not in use but marked as used (0) in map.
1 inodes are not in use but marked.
Root inode 131075 of fileset 'fset1' has been deleted.
Delete the inode reference from fileset metadata? No
Checking ACL file records
Starting Directory phase
Checking directories and files

...

Fileset 'fset1' with id 1 has corrupt record (reason 21)
Correct the fileset record? No
Fileset with id 2 is referenced in the file system
but is not recorded in the fileset metadata file.
Correct the metadata file? No

...

Error in inode 33536 snap 0: Directory block 0 has entry whose filesetId does not match
filesetId of directory
file entry inode 33537 "a"
Delete entry? No

Error in inode 33536 snap 0: Directory block 0 has entry whose filesetId does not match
filesetId of directory
file entry inode 33538 "b"
Delete entry? No

Error in inode 33536 snap 0: Directory block 0 has entry whose filesetId does not match
filesetId of directory
file entry inode 33539 "c"
Delete entry? No

...

Error in inode 3 snap 0: Directory block 0 has entry referring to a deleted inode
subdir entry inode 131075 "fset1"
Delete entry? No

Error in inode 3 snap 0: Directory block 0 has entry with invalid filesetId
subdir entry inode 33536 "dir1"
Delete entry? No

Error in inode 3 snap 0: has nlink field as 5
Correct to 3? No

...

Checking log files
Checking extended attributes file
Checking allocation summary file
Checking policy file
Validated policy '/*no policy rules*/ ':
Checking metadata of filesets
Inode space 1 does not have owning fileset.
Make fileset 'fset1' (inode -1) owner of the inode space? No
Root directory of fileset 'fset1' (inode -1) is invalid
Repair fileset root directory? No
Checking file reference counts

Error in inode 33536 snap 0: has corrupt filesetId 2
Delete file? No

Error in inode 33537 snap 0: is unreferenced
Attach inode to lost+found of fileset root filesetId 0? No

Error in inode 33538 snap 0: is unreferenced
Attach inode to lost+found of fileset root filesetId 0? No

Error in inode 33539 snap 0: is unreferenced
Attach inode to lost+found of fileset root filesetId 0? No

Error in inode 136448 snap 0: has corrupt filesetId 1
Delete file? No
```

```
Error in inode 151296 snap 0: has corrupt filesetId 1
Delete file? No
Checking file system replication status

No.   SnapId     InodeNum      FileType            Fix Error(s)                Severity
----- ---------- ------------- ------------------- --- ----------------------- -----------
1     0          38662         User                N   InodeMetadata           Harmless
2     0          33536         Directory           N   Fatal                   Noncritical
                                                       + DirectoryEntry
3     0          3             Root Directory      N   InodeMetadata           Noncritical
                                                       + DirectoryEntry
4     0          38            Fileset Metadata    N   ReservedFileRecord      Critical
5     0          33537         User                N   OrphanInode             Noncritical
6     0          33538         User                N   OrphanInode             Noncritical
7     0          33539         User                N   OrphanInode             Noncritical
8     0          136448        Directory           N   Fatal                   Noncritical
9     0          151296        Directory           N   Fatal                   Noncritical

            197120 inodes
                   59   allocated
                    3   repairable
                    0   repaired
                    3   damaged
                    0   deallocated
                    3   orphaned
                    0   attached
                    0   corrupt ACL references

              4194304 subblocks
               217682   allocated
                   14   unreferenced
                    0   duplicates
                    0   deletable
                    0   deallocated

                 1598 addresses
                    0 suspended
                    0 duplicates
                    0 reserved file holes found
                    0 reserved file holes repaired
Critical corruptions were found. Using file system without repairing the corruptions
may cause more corruptions and/or file system panics.
File system contains unrepaired damage.
Exit status 0:0:8.
mmfsck: 6027-1639 Command failed. Examine previous error messages to determine cause.
```

2. The following command checks file system `fs2`, displays inconsistencies, and makes repairs:

```
mmfsck fs2 -vy
```

The command displays output similar to the following example:

```
Checking "fs2"
FsckFlags 0x400000A
Stripe group manager <c0n0>
NeedNewLogs 0

...

Error in inode 38662 snap 0: has empty EA overflow block
Delete EA overflow block? Yes

...

Lost blocks were found.
Correct the allocation map? Yes
Block 1:332288 has map status 0x9FFFFF01 should be 0x9FFFFFFF
Block 3:332288 has map status 0x9FFFFF01 should be 0x9FFFFFFF
Checking inode map file

Corrections are needed in the inode allocation map.
Correct the allocation map? Yes
Inode 131075 is not in use but marked as used (0) in map.
1 inodes are not in use but marked.
Root inode 131075 of fileset 'fset1' has been deleted.
Delete the inode reference from fileset metadata? Yes
Checking ACL file records
Starting Directory phase
Checking directories and files
```

```
...

Fileset 'fset1' with id 1 has corrupt record (reason 21)
Correct the fileset record? Yes
Fileset with id 2 is referenced in the file system
but is not recorded in the fileset metadata file.
Correct the metadata file? Yes

...

Error in inode 33536 snap 0: Directory block 0 has entry whose filesetId does not match
filesetId of directory
file entry inode 33537 "a"
Delete entry? Yes

Error in inode 33536 snap 0: Directory block 0 has entry whose filesetId does not match
filesetId of directory
file entry inode 33538 "b"
Delete entry? Yes

Error in inode 33536 snap 0: Directory block 0 has entry whose filesetId does not match
filesetId of directory
file entry inode 33539 "c"
Delete entry? Yes

...

Error in inode 3 snap 0: Directory block 0 has entry referring to a deleted inode
subdir entry inode 131075 "fset1"
Delete entry? Yes

Error in inode 3 snap 0: Directory block 0 has entry whose filesetId does not match
filesetId of directory
subdir entry inode 33536 "dir1"
Delete entry? Yes

Error in inode 3 snap 0: has nlink field as 5
Correct to 3? Yes

...

Checking log files
Checking extended attributes file
Checking allocation summary file
Checking policy file
Validated policy '/*no policy rules*/ ':
Checking metadata of filesets
Inode space 1 does not have owning fileset.
Make fileset 'fset1' (inode -1) owner of the inode space? Yes
Root directory of fileset 'Fileset2' (inode -1) is invalid
Repair fileset root directory? Yes
The number of filesets recorded in the fileset metadata file header (2)
differs from the actual number (3).
Correct the recorded number? Yes
Checking file reference counts

Error in inode 33536 snap 0: is unreferenced
Attach inode to lost+found of fileset Fileset2 filesetId 2? Yes

Error in inode 33537 snap 0: is unreferenced
Attach inode to lost+found of fileset root filesetId 0? Yes

Error in inode 33538 snap 0: is unreferenced
Attach inode to lost+found of fileset root filesetId 0? Yes

Error in inode 33539 snap 0: is unreferenced
Attach inode to lost+found of fileset root filesetId 0? Yes

Error in inode 136448 snap 0: is unreferenced
Attach inode to lost+found of fileset fset1 filesetId 1? Yes

Error in inode 151296 snap 0: is unreferenced
Attach inode to lost+found of fileset fset1 filesetId 1? Yes
Checking file system replication status

No.   SnapId     InodeNum      FileType            Fix Error(s)                Severity
----- ---------- ------------- ------------------- --- ----------------------- -----------
1     0          38662         User                Y   InodeMetadata           Harmless
2     0          33536         Directory           Y   DirectoryEntry          Noncritical
                                                       + OrphanInode
3     0          3             Root Directory      Y   InodeMetadata           Noncritical
                                                       + DirectoryEntry
```

```
4     0          38            Fileset Metadata    Y   ReservedFileRecord      Critical
5     0          33537         User                Y   OrphanInode             Noncritical
6     0          33538         User                Y   OrphanInode             Noncritical
7     0          33539         User                Y   OrphanInode             Noncritical
8     0          136448        Directory           Y   OrphanInode             Noncritical
9     0          151296        Directory           Y   OrphanInode             Noncritical

             197120 inodes
                    61   allocated
                     4   repairable
                     4   repaired
                     0   damaged
                     0   deallocated
                     6   orphaned
                     6   attached
                     0   corrupt ACL references

               4194304 subblocks
                217682   allocated
                    14   unreferenced
                     0   duplicates
                     0   deletable
                    14   deallocated

                  1598   addresses
                     0   suspended
                     0   duplicates
                     0   reserved file holes found
                     0   reserved file holes repaired
Critical corruptions were found in the file system which were repaired.
File system is clean.
```

3. The following command checks file system FSchk, displays inconsistencies, and stores a record of each inconsistency in the patch file `path-towrite-patchfile`:

```
mmfsck FSchk -nv --patch-file path-towrite-patchfile
```

The command displays output similar to the following example:

```
Creating patch file "path-towrite-patchfile" on node "node3".
Checking "FSchk"
FsckFlags 0xA000009
Stripe group manager <c0n0>
NeedNewLogs 0

...

Error in inode 50688 snap 0: Indirect block 0 level 1 has bad disk addr at offset 21
replica 1 addr 2:318976 is a duplicate address
Delete disk address? No
Cannot fix lost blocks if not deleting duplicate address.

Error in inode 50688 snap 0: Indirect block 0 level 1 has bad disk addr at offset 20
replica 0 addr 2:318976 is a duplicate address
Delete disk address? No

Error in inode 50688 snap 0: has illReplicated field as 0
Correct to 1? No

Error in inode 50688 snap 0: has nFullBlocks field as 800
Correct to 798? No
Node 192.168.56.163 (node3) ending inode scan 0 to 32895

Error in inode 50689 snap 0: Indirect block 0 level 1 has bad disk addr at offset 40
replica 0 addr 1:371200 is a duplicate address
Delete disk address? No

Error in inode 50689 snap 0: has illReplicated field as 0
Correct to 1? No

Error in inode 50689 snap 0: has nFullBlocks field as 800
Correct to 799? No

Error in inode 50690 snap 0: Indirect block 0 level 1 has bad disk addr at offset 44
replica 0 addr 2:318976 is a duplicate address
Delete disk address? No

Error in inode 50690 snap 0: Indirect block 0 level 1 has bad disk addr at offset 792
replica 0 addr 1:371200 is a duplicate address
```

```
Delete disk address? No

Error in inode 50690 snap 0: has illReplicated field as 0
Correct to 1? No

Error in inode 50690 snap 0: has nFullBlocks field as 800
Correct to 798? No

...

Checking inode map file
Checking ACL file records
Starting Directory phase
Checking directories and files

...

Error in inode 50691 snap 0: Directory block 0 has entry with incorrect generation number
subdir entry inode 50692 "subdir7"
Delete entry? No

Error in inode 50691 snap 0: Directory block 0 has entry with incorrect generation number
subdir entry inode 13834 "subdir10"
Delete entry? No

Error in inode 50691 snap 0: has nlink field as 53
Correct to 51? No

...

Error in inode 13831 snap 0: is unreferenced
Attach inode to lost+found of fileset root filesetId 0? No

Error in inode 13834 snap 0: is unreferenced
Attach inode to lost+found of fileset root filesetId 0? No
Checking file system replication status

No.   SnapId     InodeNum      FileType            Fix Error(s)                Severity
----- ---------- ------------- ------------------- --- ----------------------- -----------
1     0          50688         User                N   InodeMetadata           Critical
                                                       + DuplicateDiskAddr
2     0          50689         User                N   InodeMetadata           Critical
                                                       + DuplicateDiskAddr
3     0          50690         User                N   InodeMetadata           Critical
                                                       + DuplicateDiskAddr
4     0          50691         Directory           N   InodeMetadata           Noncritical
                                                       + DirectoryEntry
5     0          13831         Directory           N   OrphanInode             Noncritical
6     0          13834         Directory           N   OrphanInode             Noncritical

             65792 inodes
                   98   allocated
                    4   repairable
                    0   repaired
                    0   damaged
                    0   deallocated
                    2   orphaned
                    0   attached
                    0   corrupt ACL references

              4194304   subblocks
               286532   allocated
                  160   unreferenced
                   64   duplicates
                    0   deletable
                    0   deallocated

                 1598   addresses
                    0   suspended
                    5   duplicates
                    0   reserved file holes found
                    0   reserved file holes repaired
Critical corruptions were found. Using file system without repairing the corruptions may
cause
more corruptions and/or file system panics.
File system contains unrepaired damage.
Exit status 0:0:8.
Patch file written to "node3:path-towrite-patchfile" with 21 patch entries.
mmfsck: 6027-1639 Command failed. Examine previous error messages to determine cause.
```

4. The following command uses the information in patch file `path-towrite-patchfile` to repair the file system:

```
mmfsck FSchk -v --patch-file path-towrite-patchfile --patch
```

The command displays output similar to the following example:

```
Checking "FSchk"
FsckFlags 0x1C00000A
Stripe group manager <c0n0>
NeedNewLogs 0

...

Error in inode 50688 snap 0: Indirect block 0 level 1 has bad disk addr at offset 21
replica 1 addr 2:318976 is a duplicate address
Delete disk address? Yes

Error in inode 50688 snap 0: Indirect block 0 level 1 has bad disk addr at offset 20
replica 0 addr 2:318976 is a duplicate address
Delete disk address? Yes

Error in inode 50688 snap 0: has illReplicated field as 0
Correct to 1? Yes

Error in inode 50688 snap 0: has nFullBlocks field as 800
Correct to 798? Yes

Error in inode 50689 snap 0: Indirect block 0 level 1 has bad disk addr at offset 40
replica 0 addr 1:371200 is a duplicate address
Delete disk address? Yes

Error in inode 50689 snap 0: has illReplicated field as 0
Correct to 1? Yes

Error in inode 50689 snap 0: has nFullBlocks field as 800
Correct to 799? Yes

Error in inode 50690 snap 0: Indirect block 0 level 1 has bad disk addr at offset 44
replica 0 addr 2:318976 is a duplicate address
Delete disk address? Yes

Error in inode 50690 snap 0: Indirect block 0 level 1 has bad disk addr at offset 792
replica 0 addr 1:371200 is a duplicate address
Delete disk address? Yes

Error in inode 50690 snap 0: has illReplicated field as 0
Correct to 1? Yes

Error in inode 50690 snap 0: has nFullBlocks field as 800
Correct to 798? Yes

Lost blocks were found.
Correct the allocation map? Yes
Block 1:134144 has map status 0x00000000 should be 0xFFFFFFFF
Block 3:134656 has map status 0x00000000 should be 0xFFFFFFFF
Block 2:239104 has map status 0x00000000 should be 0xFFFFFFFF
Block 2:318976 has map status 0x00000000 should be 0xFFFFFFFF
Block 1:371200 has map status 0x00000000 should be 0xFFFFFFFF

Error in inode 50691 snap 0: Directory block 0 has entry with incorrect generation number
subdir entry inode 50692 "subdir7"
Delete entry? Yes

Error in inode 50691 snap 0: Directory block 0 has entry with incorrect generation number
subdir entry inode 13834 "subdir10"
Delete entry? Yes

Error in inode 50691 snap 0: has nlink field as 53
Correct to 51? Yes
52 % complete on Tue Feb 21 13:34:53 2017
Scanning patch file for post-mount patches

Error in inode 13831 snap 0: is unreferenced
Attach inode to lost+found of fileset root filesetId 0? Yes

Error in inode 13834 snap 0: is unreferenced
Attach inode to lost+found of fileset root filesetId 0? Yes
100 % complete on Tue Feb 21 13:34:53 2017
```

```
No.   SnapId     InodeNum      FileType            Fix Error(s)                 Severity
----- ---------- ------------- ------------------- --- ----------------------- -----------
1     0          50688         User                Y   InodeMetadata           Critical
                                                       + DuplicateDiskAddr
2     0          50689         User                Y   InodeMetadata           Critical
                                                       + DuplicateDiskAddr
3     0          50690         User                Y   InodeMetadata           Critical
                                                       + DuplicateDiskAddr
4     0          50691         Directory           Y   InodeMetadata           Noncritical
                                                       + DirectoryEntry
5     0          13831         Directory           Y   OrphanInode             Noncritical
6     0          13834         Directory           Y   OrphanInode             Noncritical
Critical corruptions were found in the file system which were repaired.
File system is patched.
```

5. The following command checks file system fs1 in online mode and detects corruptions in the block allocation map.

```
mmfsck fs1 -v -n -o

Checking "fs1" on Tue Aug 17 18:10:40 2021
  FsckFlags                     0x2000099
  FsckFlags2                    0x0
  FsckFlags3                    0x0
  Stripe group manager          192.168.56.164 (node4)
  Is offline fsck running       0
  NeedNewLogs                   0
…
Lost block   2:010523392  map status is 0x00000000  should be 0xFFFFFFFF

Lost blocks were found.
Correct the allocation map? No
….
InodeProblemList: 1 entries
iNum          snapId     status keep delete noScan new error
------------ ---------- ------ ---- ------ ------ --- ------------------
          38          0      3    0      0      0   1 0x00000050 AddrCorrupt RepAddrCorrup

Error in block allocation map file inode 38:  AddrCorrupt RepAddrCorrup

 100 % complete on Tue Aug 17 18:10:43 2021

             15974400 subblocks
               223913   allocated
                   32   unreferenced
                    0   deallocated

                 3409 addresses
                    0   suspended

InodeProblemList: 1 entries
iNum          snapId     status keep delete noScan new error
------------ ---------- ------ ---- ------ ------ --- ------------------
          38          0      3    0      0      0   0 0x00000050 AddrCorrupt RepAddrCorrup
File system contains unrepaired damage.
Fsck completed in 0 hours 0 minutes 4 seconds
Exit status 0:10:8.
```

See also

- “mmcheckquota command” on page 220
- “mmcrfs command” on page 318
- “mmdelfs command” on page 373
- “mmdf command” on page 387
- “mmlsfs command” on page 506

Location

`/usr/lpp/mmfs/bin`

mmfsctl command

Issues a file system control request.

Synopsis

```
mmfsctl Device {suspend | suspend-write | resume}
```

or

```
mmfsctl Device {exclude | include}
              {-d "DiskName[;DiskName...]" | -F DiskFile | -G FailureGroup}
```

or

```
mmfsctl Device syncFSconfig
              {-n RemoteNodesFile | -C RemoteClusterName} [-S SpecFile]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmfsctl` command to issue control requests to a particular GPFS file system. The command is used to temporarily suspend the processing of all application I/O requests, and later resume them, as well as to synchronize the file system's configuration state between peer clusters in disaster recovery environments.

See *Establishing disaster recovery for your GPFS cluster* in *IBM Spectrum Scale: Administration Guide*.

Using `mmfsctl suspend` and `mmfsctl resume`

Before creating a FlashCopy® image of the file system, the user must run `mmfsctl suspend` to temporarily quiesce all file system activity and flush the internal buffers on all nodes that mount this file system. The on-disk metadata will be brought to a consistent state, which provides for the integrity of the FlashCopy snapshot. If a request to the file system is issued by the application after the invocation of this command, GPFS suspends this request indefinitely, or until the user issues `mmfsctl resume`.

Once the FlashCopy image has been taken, the `mmfsctl resume` command can be issued to resume the normal operation and complete any pending I/O requests.

Using `mmfsctl syncFSconfig`

The `mmfsctl syncFSconfig` command extracts the file system's related information from the local GPFS configuration data, transfers this data to one of the nodes in the peer cluster, and attempts to import it there.

Once the GPFS file system has been defined in the primary cluster, users run this command to import the configuration of this file system into the peer recovery cluster. After producing a FlashCopy image of the file system and propagating it to the peer cluster using Peer-to-Peer Remote Copy (PPRC), users similarly run this command to propagate any relevant configuration changes made in the cluster after the previous snapshot.

The primary cluster configuration server of the peer cluster must be available and accessible using remote shell and remote copy at the time of the invocation of the `mmfsctl syncFSconfig` command, and remote nodes must be reachable by the ping utility. Also, the peer GPFS clusters should be defined to use the same remote shell and remote copy mechanism, and they must be set up to allow nodes in peer clusters to communicate without the use of a password.

Note: In a cluster that is CCR-enabled, you cannot run `mmfsctl syncFSconfig` on a file system that has tiebreaker disks.

Not all administrative actions performed on the file system necessitate this type of resynchronization. It is required only for those actions that modify the file system information maintained in the local GPFS configuration data. These actions include:

- Adding, removing, and replacing disks (commands `mmadddisk`, `mmdeldisk`, `mmrpldisk`)
- Modifying disk attributes (command `mmchdisk`)
- Changing the file system's mount point (command `mmchfs -T`)
- Changing the file system device name (command `mmchfs -W`)

The process of synchronizing the file system configuration data can be automated by utilizing the `syncfsconfig` user exit.

Using `mmfsctl exclude`

The `mmfsctl exclude` command is to be used only in a disaster recovery environment, only after a disaster has occurred, and only after ensuring that the disks in question have been physically disconnected. Otherwise, unexpected results may occur.

The `mmfsctl exclude` command can be used to manually override the file system descriptor quorum after a site-wide disaster. See *Establishing disaster recovery for your GPFS cluster* in *IBM Spectrum Scale: Administration Guide*. This command enables users to restore normal access to the file system with less than a quorum of available file system descriptor replica disks, by effectively excluding the specified disks from all subsequent operations on the file system descriptor. After repairing the disks, the `mmfsctl include` command can be issued to restore the initial quorum configuration.

Parameters

Device

The device name of the file system. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**. If `all` is specified with the `syncFSconfig` option, this command is performed on all GPFS file systems defined in the cluster.

The following options can be specified after *Device*:

suspend
Instructs GPFS to flush the internal buffers on all nodes, bring the file system to a consistent state on disk, and suspend the processing of all subsequent application I/O requests.

suspend-write
Suspends the execution of all new write I/O requests coming from user applications, flushes all pending requests on all nodes, and brings the file system to a consistent state on disk.

resume
Instructs GPFS to resume the normal processing of I/O requests on all nodes.

exclude
Instructs GPFS to exclude the specified group of disks from all subsequent operations on the file system descriptor, and change their availability state to down, if the conditions in the following Note are met.

If necessary, this command assigns additional disks to serve as the disk descriptor replica holders, and migrate the disk descriptor to the new replica set. The excluded disks are not deleted from the file system, and still appear in the output of the `mmlsdisk` command.

Note: The `mmfsctl exclude` command is to be used only in a disaster recovery environment, only after a disaster has occurred, and only after ensuring that the disks in question have been physically disconnected. Otherwise, unexpected results may occur.

include
Informs GPFS that the previously excluded disks have become operational again. This command writes the up-to-date version of the disk descriptor to each of the specified disks, and clears the `excl` tag.

-d "*DiskName*[;*DiskName*...]"
Specifies the names of the NSDs to be included or excluded by the `mmfsctl` command. Separate the names with semicolons (;) and enclose the list of disk names in quotation marks.

-F *DiskFile*
Specifies a file containing the names of the NSDs, one per line, to be included or excluded by the `mmfsctl` command.

-G *FailureGroup*
A failure group identifier for the disks to be included or excluded by the `mmfsctl` command.

syncFSconfig
Synchronizes the configuration state of a GPFS file system between the local cluster and its peer in two-cluster disaster recovery configurations.

The following options can be specified after `syncFSconfig`:

-n *RemoteNodesFile*
Specifies a list of contact nodes in the peer recovery cluster that GPFS uses when importing the configuration data into that cluster. Although any node in the peer cluster can be specified here, users are advised to specify the identities of the peer cluster's primary and secondary cluster configuration servers, for efficiency reasons.

-C *RemoteClusterName*
Specifies the name of the GPFS cluster that owns the remote GPFS file system.

-S *SpecFile*
Specifies the description of changes to be made to the file system, in the peer cluster during the import step. The format of this file is identical to that of the *ChangeSpecFile* used as input to the `mmimportfs` command. This option can be used, for example, to define the assignment of the NSD servers for use in the peer cluster.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Results

The `mmfsctl` command returns 0 if successful.

Security

You must have root authority to run the `mmfsctl` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

This sequence of commands creates a FlashCopy image of the file system and propagates this image to the recovery cluster using the Peer-to-Peer Remote Copy technology. The following configuration is assumed:

Site	LUNs
Primary cluster (site A)	lunA1, lunA2
Recovery cluster (site B)	lunB1

lunA1
FlashCopy source

lunA2
FlashCopy target, PPRC source

lunB1
PPRC target

A single GPFS file system named `fs0` has been defined in the primary cluster over lunA1.

1. In the primary cluster, suspend all file system I/O activity and flush the GPFS buffers

   ```
   mmfsctl fs0 suspend
   ```

   The output is similar to this:

   ```
   Writing dirty data to disk.
   Quiescing all file system operations.
   Writing dirty data to disk again.
   ```

2. Establish a FlashCopy pair using lunA1 as the source and lunA2 as the target.
3. Resume the file system I/O activity:

   ```
   mmfsctl fs0 resume
   ```

   The output is similar to this:

   ```
   Resuming operations.
   ```

4. Establish a Peer-to-Peer Remote Copy (PPRC) path and a synchronous PPRC volume pair lunA2-lunB1 (primary-secondary). Use the 'copy entire volume' option and leave the 'permit read from secondary' option disabled.
5. Wait for the completion of the FlashCopy background task. Wait for the PPRC pair to reach the duplex (fully synchronized) state.
6. Terminate the PPRC volume pair lunA2-lunB1.
7. If this is the first time the snapshot is taken, or if the configuration state of `fs0` changed since the previous FlashCopy snapshot, propagate the most recent configuration to site B:

   ```
   mmfsctl fs0 syncFSconfig -n recovery_clust_nodelist
   ```

Location

`/usr/lpp/mmfs/bin`

mmgetacl command

Displays the GPFS access control list of a file or directory.

Synopsis

```
mmgetacl [-d] [-o OutFilename] [-k {nfs4 | posix | native}] Filename
```

Availability

Available on all IBM Spectrum Scale editions. Available on AIX and Linux.

Description

Use the `mmgetacl` command to display the ACL of a file or directory.

For information about NFS V4 ACLs, see the topics *Managing GPFS access control lists* and *NFS and GPFS* in the *IBM Spectrum Scale: Administration Guide*.

Users may need to see ACLs in their true form as well as how they are translated for access evaluations. There are four cases:

1. By default, `mmgetacl` returns the ACL in a format consistent with the file system setting, specified using the `-k` flag on the `mmcrfs` or `mmchfs` commands.

   If the setting is `posix`, the ACL is shown as a traditional ACL.

   If the setting is `nfs4`, the ACL is shown as an NFS V4 ACL.

   If the setting is `all`, the ACL is returned in its true form.
2. The command `mmgetacl -k nfs4` always produces an NFS V4 ACL.
3. The command `mmgetacl -k posix` always produces a traditional ACL.
4. The command `mmgetacl -k native` always shows the ACL in its true form regardless of the file system setting.

The following describes how `mmgetacl` works for POSIX and NFS V4 ACLs:

```
Command               ACL    mmcrfs -k  Display        -d (default)
-------------------   -----  ---------  -------------  --------------
mmgetacl              posix  posix      Access ACL     Default ACL
mmgetacl              posix  nfs4       NFS V4 ACL     Error[1]
mmgetacl              posix  all        Access ACL     Default ACL
mmgetacl              nfs4   posix      Access ACL[2]  Default ACL[2]
mmgetacl              nfs4   nfs4       NFS V4 ACL     Error[1]
mmgetacl              nfs4   all        NFS V4 ACL     Error[1]
mmgetacl -k native    posix  any        Access ACL     Default ACL
mmgetacl -k native    nfs4   any        NFS V4 ACL     Error[1]
mmgetacl -k posix     posix  any        Access ACL     Default ACL
mmgetacl -k posix     nfs4   any        Access ACL[2]  Default ACL[2]
mmgetacl -k nfs4      any    any        NFS V4 ACL     Error[1]
------------------------------------------------------------------------
[1] NFS V4 ACLs include inherited entries. Consequently, there cannot
    be a separate default ACL.
[2] Only the mode entries (owner, group, everyone) are translated.
    The rwx values are derived from the
    NFS V4 file mode attribute. Since the NFS V4 ACL is more granular
    in nature, some information is lost in this translation.
------------------------------------------------------------------------
```

Parameters

Filename
The path name of the file or directory for which the ACL is to be displayed. If the `-d` option is specified, *Filename* must contain the name of a directory.

Options

-d
Specifies that the default ACL of a directory is to be displayed.

-k {nfs4 | posix | native}

nfs4
Always produces an NFS V4 ACL.

posix
Always produces a traditional ACL.

native
Always shows the ACL in its true form regardless of the file system setting.

-o *OutFilename*
The path name of a file to which the ACL is to be written.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have read access to the directory where the file exists to run the `mmgetacl` command.

You may issue the `mmgetacl` command only from a node in the GPFS cluster where the file system is mounted.

Examples

1. To display the ACL for a file named `project2.history`, issue this command:

   ```
   mmgetacl project2.history
   ```

   The system displays information similar to:

   ```
   #owner:paul
   #group:design
   user::rwxc
   group::r-x-
   other::r-x-
   ```

2. This is an example of an NFS V4 ACL displayed using `mmgetacl`. Each entry consists of three lines reflecting the greater number of permissions in a text format. An entry is either an `allow` entry or a `deny` entry. An X indicates that the particular permission is selected, a minus sign (–) indicates that is it not selected. The following access control entry explicitly allows READ, EXECUTE and READ_ATTR to the `staff` group on a file:

   ```
   group:staff:r-x-:allow
    (X)READ/LIST (-)WRITE/CREATE (-)APPEND/MKDIR (-)SYNCHRONIZE (-)READ_ACL  (X)READ_ATTR
   (-)READ_NAMED
    (-)DELETE    (-)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL (-)WRITE_ATTR
   (-)WRITE_NAMED
   ```

3. This is an example of a directory ACLs, which may include *inherit* entries (the equivalent of a default ACL). These do not apply to the directory itself, but instead become the initial ACL for any

objects created within the directory. The following access control entry explicitly denies READ/LIST, READ_ATTR, and EXEC/SEARCH to the sys group.

```
group:sys:----:deny:DirInherit
 (X)READ/LIST (-)WRITE/CREATE (-)APPEND/MKDIR (-)SYNCHRONIZE (-)READ_ACL  (X)READ_ATTR
(-)READ_NAMED
 (-)DELETE    (-)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL (-)WRITE_ATTR
(-)WRITE_NAMED
```

See also

- “mmeditacl command” on page 401
- “mmdelacl command” on page 360
- “mmputacl command” on page 617

Location

/usr/lpp/mmfs/bin

mmgetstate command

Displays the state of the GPFS daemon on one or more nodes.

Synopsis

```
mmgetstate [-L] [-s] [-v] [-Y] [-a | -N {Node[,Node...] | NodeFile | NodeClass}]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmgetstate` command displays the state of the GPFS daemon on the specified nodes.

Parameters

-a
Displays the state of the GPFS daemon on all nodes in the cluster.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Displays the state of the GPFS daemon information on the specified nodes.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

This command does not support a *NodeClass* of `mount`.

Options

-L
Displays the number of quorum nodes, the number of nodes that are up, the total number of nodes, the state of the GPFS daemon, and other information.

-s
Displays summary information, such as the number of local and remote nodes that are joined in the cluster and the number of quorum nodes.

-v
Displays intermediate error messages.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

The command recognizes and displays the following GPFS states:

active
The GPFS daemon is ready for operations.

arbitrating
A node is trying to form a quorum with the other available nodes.

down
The GPFS daemon is not running on the node or is recovering from an internal error.

unknown
An unknown value. The command cannot connect with the node or some other error occurred.

unresponsive
The GPFS daemon is running but is not responding.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmgetstate` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To display the quorum, the number of nodes up, and the total number of nodes for the GPFS cluster, issue the following command:

```
mmgetstate -a -L
```

The system displays output similar to the following example:

```
Node number Node name Quorum Nodes up Total nodes GPFS state Remarks
------------------------------------------------------------------------------
       1    c5n92        3       5       12       active
       2    c5n94        3       5       12       active
       3    c5n95        3       5       12       active     quorum node
       4    c5n96        3       5       12       active
       5    c5n97        3       5       12       active     quorum node
       6    c5n98        3       5       12       active
       7    c5n107       3       5       12       active     quorum node
       8    c5n108       3       5       12       active
       9    c5n109       3       5       12       active     quorum node
      10    c5n110       3       5       12       down
      11    c5n111       3       5       12       active     quorum node
      12    c5n112       3       5       12       active
```

The 3 under the Quorum column means that you must have three quorum nodes up to achieve quorum.

2. In the following example, the cluster uses node quorum with tiebreaker disks. The asterisk (*) in the Quorum field indicates that tiebreaker disks are being used:

```
mmgetstate -a -L
```

The system displays output similar to the following example:

```
Node number Node name Quorum Nodes up Total nodes GPFS state Remarks
------------------------------------------------------------------------------
     1      k5n91        5*      8       21       active
     2      k5n92        5*      8       21       active    quorum node
     3      k5n94        5*      8       21       active
     5      k5n96        5*      8       21       active
     6      k5n97        5*      8       21       active    quorum node
     7      k5n98        5*      8       21       active
     8      k5n99        5*      8       21       active    quorum node
```

3. To display summary information, issue this command:

```
mmgetstate -s
```

The system displays output similar to the following example:

```
Node number  Node name  GPFS state
------------------------------------

           1  c5n90      active


Summary information
---------------------

Number of nodes defined in the cluster:            6
Number of local nodes active in the cluster:       6
Number of remote nodes joined in this cluster:     0
Number of quorum nodes defined in the cluster:     6
Number of quorum nodes active in the cluster:      6
Quorum = 4, Quorum achieved
```

See also

- “mmchconfig command” on page 170
- “mmcrcluster command” on page 306
- “mmshutdown command” on page 706
- “mmstartup command” on page 726

Location

/usr/lpp/mmfs/bin

mmhadoopctl command

Installs and sets up the GPFS connector for a Hadoop distribution; starts or stops the GPFS connector daemon on a node.

Synopsis

```
mmhadoopctl connector {start |stop |getstate}
```

or

```
mmhadoopctl connector syncconf [--nocheck] [path]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmhadoopctl` command to install and set up the GPFS connector for a Hadoop distribution, or to start or stop the GPFS connector daemon on a node.

Parameters

connector
Controls the GPFS connector daemon with one of the following actions:

start
Starts the connector daemon.

stop
Stops the connector daemon.

getstate
Detects whether the connector daemon is running and shows its process ID.

syncconf
Synchronizes the connector configuration in the cluster.

--nocheck
If --nocheck is used, the sanity check is not performed. Therefore, use this option with caution.

path
Specifies the path to the config directory or file.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmhadoopctl` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

To start the GPFS connector daemon, issue this command:

```
mmhadoopctl connector start
```

The system displays output similar to this:

```
Hadoop connector 'gpfs-connector-daemon' started
```

Location

/usr/lpp/mmfs/bin

mmhdfs command

The **mmhdfs** command configures and manages the IBM Spectrum Scale HDFS Transparency components.

Synopsis

```
mmhdfs {hdfs | hdfs-nn | hdfs-dn} {start | stop | restart | status}
```

or

```
mmhdfs {namenode | datanode} {start | stop | restart | status} [-N Node[,Node...]]
```

or

```
mmhdfs config upload
```

or

```
mmhdfs config set {filename} {-k key1=value1[ -k key2=value2...]}
```

or

```
mmhdfs config del {filename} {-k key1[ -k key2...]}
```

or

```
mmhdfs config get {filename} {-k key1[ -k key2...]}
```

or

```
mmhdfs config import [--nocheck] {dirpath} {filename1[,filename2...] | all}
```

or

```
mmhdfs config export [--nocheck] {dirpath} {filename1[,filename2...] | all}
```

or

```
mmhdfs worker {add | remove} [Node[,Node...]]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Note: From IBM Spectrum Scale 5.0.4.2, the **mmhdfs** command is available when HDFS Transparency 3.1.1 or later is installed.

Use the **mmhdfs** command to perform the following:

- Start and Stop HDFS Transparency.
- Add and Remove the HDFS Transparency worker nodes.
- Manage the HDFS Transparency configurations.

Parameters

hdfs
Specifies the NameNode and DataNode service of all the HDFS Transparency nodes.

hdfs-nn
Specifies the NameNode service of all the NameNodes.

hdfs-dn
Specifies the DataNode service of all the DataNodes.

start
Starts the specified component.

stop
Stops the specified component.

restart
Restarts the specified component.

status
Shows the running status of the specified component.

namenode
Specifies the NameNode service of the specified nodes.

datanode
Specifies the DataNode service of the specified nodes.

start
Starts the specified component.

stop
Stops the specified component.

restart
Restarts the specified component.

status
Shows the running status of the specified component.

-N *Node*[,*Node*...]
Specifies the nodes on which the command should run.

config
Specifies the configuration operation.

upload
Uploads the configuration into CCR.

set
Sets the specified config option of the specified configuration file to the specified value.

filename
Specifies the name of the file.

-k key1=*value1*
Specifies the key value pair to set.

del
Deletes the specified config option from the specified configuration file.

filename
Specifies the name of the configuration file.

-k key1
Specifies the key to delete.

get
Gets the value of the specified config option for the specified configuration file.

filename
Specifies the name of the configuration file.

-k key1
Specifies the key to delete.

import
Imports the specified configuration files from the specified directory.

--nocheck
If this option is not specified, only the following configuration files are supported:

hadoop-env.sh, core-site.xml, hadoop-policy.xml, hdfs-site.xml, httpfs-site.xml, gpfs-site.xml, kms-acls.xml, kms-site.xml, ranger-hdfs-audit.xml, ranger-hdfs-security.xml, ranger-policymgr-ssl.xml, ranger-security.xml, ssl-client.xml, ssl-server.xml, yarn-site.xml, kms-log4j.properties, log4j.properties, hadoop-metrics2.properties, hadoop-metrics.properties and workers.

dirpath
Specifies the directory from which to import the files.

filename
Configuration files to be imported.

all
Import all configuration files from the directory.

export
Exports the specified configuration files to the specified directory.

--nocheck
If this option is not specified, only the following configuration files are supported:

hadoop-env.sh, core-site.xml, hadoop-policy.xml, hdfs-site.xml, httpfs-site.xml, gpfs-site.xml, kms-acls.xml, kms-site.xml, ranger-hdfs-audit.xml, ranger-hdfs-security.xml, ranger-policymgr-ssl.xml, ranger-security.xml, ssl-client.xml, ssl-server.xml, yarn-site.xml, kms-log4j.properties, log4j.properties, hadoop-metrics2.properties, hadoop-metrics.properties and workers.

dirpath
Specifies the directory from which to export the files.

filename
Configuration files to be exported.

all
Export all configuration files to the directory.

worker
Adds or removes the DataNode(s).

add
Adds the specified nodes as DataNodes.

remove
Removes the specified nodes from the DataNodes.

Node[,Node...]
Specifies a comma-delimited list of the nodes on which the command should run.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the **mmhdfs** command.

The node on which the command is issued must be able to execute the remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

Examples

1. To start all the HDFS Transparency components, run the following command:

```
mmhdfs hdfs start
```

2. To stop all the HDFS Transparency components, run the following command:

```
mmhdfs hdfs stop
```

3. To check the HDFS Transparency status for all the Namenodes, run the following command:

```
mmhdfs hdfs-nn status
```

4. To add in a configuration setting, run the following command:

```
mmhdfs config set hadoop-env.sh -k JAVA_HOME=/usr/jdk64/jdk1.8.0_112
```

or

```
mmhdfs config set gpfs-site.xml -k gpfs.mnt.dir=/gpfs -k gpfs.data.dir=trunk -k
gpfs.storage.type=local -k gpfs.replica.enforced=gpfs
```

5. To get a configuration setting, run the following command:

```
mmhdfs config get hadoop-env.sh -k JAVA_HOME JAVA_HOME=/usr/jdk64/jdk1.8.0_112
```

or

```
mmhdfs config get core-site.xml -k fs.defaultFS fs.defaultFS=hdfs://hdfscluster
```

6. To delete a configuration setting, run the following command:

```
mmhdfs config del hadoop-env.sh -k JAVA_HOME
```

or

```
mmhdfs config del core-site.xml -k fs.defaultFS
```

7. To import configuration files from a local directory, run the following command:

```
mmhdfs config import /tmp/hadoop core-site.xml
```

8. To import all configuration files from a local directory, run the following command:

```
mmhdfs config import /tmp/hadoop all
```

9. To export configuration files to a local directory, run the following command:

```
mmhdfs config export /tmp/hadoop core-siste.xml
```

10. To export all configuration files to a local directory, run the following command:

```
mmhdfs config export /tmp/hadoop all
```

11. To add Datanodes into the HDFS Transparency cluster, run the following command:

```
mmhdfs worker add c902f05x05, c902f05x06, c902f05x07
```

12. To upload configuration files into CCR and other nodes, run the following command:

```
mmhdfs config upload
```

See also

- *CES HDFS* topic in *IBM Spectrum Scale: Big Data and Analytics Guide*
- *Installing IBM Spectrum Scale on Linux nodes and deploying protocols* topic in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*
- *Configuring with the installation toolkit* topic in *IBM Spectrum Scale: Administration Guide*
- “mmces command” on page 133
- “mmlsfs command” on page 506

Location

`/usr/lpp/mmfs/hadoop/sbin`

mmhealth command

Monitors health status of nodes.

Synopsis

```
mmhealth node show [component]
                   [-N {Node[,Node..] | NodeFile | NodeClass}]
                   [-Y] [--verbose] [--unhealthy]
                   [--color | --nocolor]
                   [--resync]
```

or

```
mmhealth node eventlog [[--hour | --day | --week | --month] | [--clear] | [--verbose]]
                       [--show-state-changes][-N {Node[,Node..] | NodeFile | NodeClass}]
                       [-Y][--color | --nocolor]
```

or

```
mmhealth event show {EventName | EventID} [-N {Node[,Node..] | NodeFile | NodeClass}]
```

or

```
mmhealth event hide EventName [Entity_Name]
```

or

```
mmhealth event resolve EventName [Entity_Name]
```

or

```
mmhealth event unhide EventName [Entity_Name]
```

or

```
mmhealth event list HIDDEN
```

or

```
mmhealth cluster show [component]
                      [-Y][--verbose][--unhealthy]
                      [--color | --nocolor]
```

or

```
mmhealth thresholds list [--verbose]
```

or

```
mmhealth thresholds add { metric [: sum | avg | min | max | rate ]| measurement
                          [--errorlevel {threshold error limit}
                        [--warnlevel{threshold warn limit }]
                        |--direction { high|low }]
                        [--sensitivity {bucketsize } ] [--hysteresis { percentage }]
                        [--filterBy] [--groupBy ] [--name { ruleName }]
                        [--errormsg {user defined action description}]
                        [--warnmsg {user defined action description}]
```

or

```
mmhealth thresholds delete {ruleName | all }
```

or

```
mmhealth config interval [OFF | LOW | MEDIUM | DEFAULT | HIGH]
```

or

```
mmhealth config monitor [-Y][-N {Node[,Node..] | NodeFile | NodeClass}][pause | resume]
```

or

```
mmhealth config webhook [ list [ -Y ] |  add <URL>  | remove <URL> ]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the **mmhealth** command to monitor the health of the node and services hosted on the node in IBM Spectrum Scale.

The IBM Spectrum Scale administrator can monitor the health of each node and the services hosted on that node using the **mmhealth** command. The **mmhealth** command also shows the events that are responsible for the unhealthy status of the services hosted on the node. This data can be used to monitor and analyze the reasons for the unhealthy status of the node. The **mmhealth** command acts as a problem determination tool to identify which services of the node are unhealthy, and find the events responsible for the unhealthy state of the service.

The **mmhealth** command also monitors the state of all the IBM Spectrum Scale RAID components such as array, physical disk, virtual disk, and enclosure of the nodes that belong to the recovery group.

For more information about the system monitoring feature, see the *Monitoring system health by using the mmhealth command* section in the *IBM Spectrum Scale: Problem Determination Guide*.

The **mmhealth** command shows the details of threshold rules. This detail helps to avoid out-of-space errors for file systems. The space availability of the file system component depends upon the occupancy level of fileset-inode spaces and the capacity usage in each data or metadata pool. The violation of any single rule triggers the parent file system's capacity-issue events. The capacity metrics are frequently compared with the rules boundaries by internal monitor process. If any of the metric values exceeds their threshold limit, then the system health (daemon/service) receives an event notification from monitor process and generate a RAS event for the file system for space issues. For the predefined capacity utilization rules, the warning level is set to 80%, and the error level to 90%. For memory utilization rule, the warn level is set to 100 MB, and the error level to 50 MB. You can use the **mmlsfileset** and the **mmlspool** commands to track the inode and pool space usage.

Parameters

node
Displays the health status, specifically, at node level.

show
Displays the health status of the specified component.

[*component*]
The value of the component can be one of the following:

AFM | ARRAY | AUTH | AUTH_OBJ | BLOCK | CALLHOME | CANISTER | CES | CESIP | CESNETWORK | CLOUDGATEWAY | DISK | ENCLOSURE | ENCRYPTION | FILEAUDITLOG | FILESYSTEM | FILESYSMGR | GPFS | GUI | HADOOPCONNECTOR | HDFS_DATANODE | HDFS_NAMENODE | LOCALCACHE | MSGQUEUE | NATIVE_RAID | NETWORK | NFS | NODE

| NVME | OBJECT | PERFMON | PHYSICALDISK | RECOVERY GROUP | SMB | THRESHOLD | VIRTUALDISK | WATCHFOLDER

Displays the detailed health status of the specified component.

The following components are specific to Elastic Storage Server (ESS):

- Native RAID
- Array
- Physical disk
- Virtual disk
- Recovery group
- Enclosure

For more information on these components, see Elastic Storage Server.

Important:

The HADOOPCONNECTOR component is used only in old deployments, where HDFS is configured outside the CES, while the HDFS_DATANODE and the HDFS_NAMENODE components are used when HDFS is configured as a CES service.

The canister component is specific to the IBM Elastic Storage® System. For more information about this component, see the *Canister events* section in the IBM Elastic Storage System: Problem Determination Guide.

UserDefinedSubComponent

Displays services that are named by the customer, categorized by one of the other hosted services. For example, a file system named gpfs0 is a subcomponent of file system.

-N

Allows the system to make remote calls to the other nodes in the cluster for:

Node[,Node....]

Specifies the node or list of nodes for which the health status is displayed.

NodeFile

Specifies a file containing a list of node descriptors, one per line, for which the health status is displayed.

NodeClass

Specifies the node class for which the health status is displayed.

-Y

Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

--verbose

Shows the detailed health status of a node, including its sub-components.

--unhealthy

Displays the unhealthy components only.

--color | --nocolor

If you run the command without --color or --nocolor, the system automatically detects if a tty is attached and uses the color mode.

If the --color is specified, it uses the color output even if no tty is attached.

If the --nocolor is specified, it uses the non-colored output.

--resync

Use this option to resynch all the health states and events of the current node with the cluster state manager. The cluster state manager collects the cluster wide health data.

eventlog

Shows the event history for a specified period of time. If no time period is specified, it displays all the events by default:

-Y

Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

[--hour | --day | --week| --month]

Displays the event history for the specified time period.

[--clear]

Clears the event log's database. This action cannot be reversed.

CAUTION: The events database is used by the mmhealth node eventlog as well as the mmces events list. If you clear the database, it also affects the mmces events list. Ensure that you use the **--clear** option with caution.

[--verbose]

Displays additional information about the event like component name and event ID in the eventlog.

[--show-state-changes]

Displays additional information about the changes in the state of the components of a node.

Note: This option is only valid if the cluster has a minimum release level of 5.0.2.

--color | --nocolor

If you run the command without --color or --nocolor, it automatically detects if a tty is attached and uses the color mode.

If the --color is specified, it uses the color output even if no tty is attached.

If the --nocolor is specified, it uses the non-colored output.

event

Gives the details of various events:

show

Shows the detailed description of the specified event:

EventName

Displays the detailed description of the specified event name.

EventID

Displays the detailed description of the specified event ID.

hide

Hides the specified TIP events.

unhide

Reveals the TIP events that were previously hidden using the `hide`.

resolve

Manually resolve error events. All events starting with `fserr*` can be manually resolved with this command. The only other event that can be manually resolved is `out_of_memory`.

list HIDDEN

Shows all the TIP events that are added to the list of hidden events.

cluster
Displays the health status of all nodes and monitored node components in the cluster.

show
Displays the health status of the specified component.

[*component*]
The value of the component can be one of the following:

AFM | ARRAY | AUTH | AUTH_OBJ | BLOCK | CALLHOME | CANISTER | CES | CESIP | CESNETWORK | CLOUDGATEWAY | DISK | ENCLOSURE | ENCRYPTION | FILEAUDITLOG | FILESYSTEM | FILESYSMGR | GPFS | GUI | HADOOPCONNECTOR | HDFS_DATANODE | HDFS_NAMENODE | LOCALCACHE | MSGQUEUE | NATIVE_RAID | NETWORK | NFS | NODE | NVME | OBJECT | PERFMON | PHYSICALDISK | RECOVERY GROUP | SMB | THRESHOLD | VIRTUALDISK | WATCHFOLDER
Displays the detailed health status of the specified component.

The following components are specific to Elastic Storage Server (ESS):

- Native RAID
- Array
- Physical disk
- Virtual disk
- Recovery group
- Enclosure

For more information on these components, see Elastic Storage Server.

Important:

The HADOOPCONNECTOR component is used only in old deployments where HDFS is configured outside the CES, while the HDFS_DATANODE and the HDFS_NAMENODE components are used when HDFS is configured as a CES service.

The canister component is specific to the IBM Elastic Storage System. For more information about this component, see the *Canister events* section in the IBM Elastic Storage System: Problem Determination Guide.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

--verbose
Shows the detailed health status of a node, including its sub-components.

--unhealthy
Displays the unhealthy components only.

--color | --nocolor

If you run the command without --color or --nocolor, it automatically detects if a tty is attached and uses the color mode.

If the --color is specified, it uses the color output even if no tty is attached.

If the --nocolor is specified, it uses the non-colored output.

thresholds list
Displays the list of the threshold rules defined for the system.

--verbose

Shows all attributes of each threshold rule, including its running state.

Note: A threshold rule can be in one of the following states:

Active
A rule is running.

Inactive
A rule is not running. Either no keys for the defined metric measurement is found in the performance monitoring tool metadata or the corresponding sensor is not enabled. As soon as the metric keys and metric values are detected for a rule, the state of the rule switches to `active`.

Unknown
The state of a rule could not be determined. Probably an issue with querying internal data.

thresholds add
Creates a new thresholds rule for the specified metric or measurement, and activates monitoring process stores for this rule.

Note: A measurement is a value calculated using more than one metric in a pre-defined formula.

metric [: SUM | AVG | MIN | MAX | RATE]
Creates a threshold for the specified metric. All metrics that are supported by the performance monitoring tool, and use raw values or are downsampled by aggregators (sum, avg, min, max, rate) can be used. For a list of metrics supported by the performance monitoring tool, see the *List of performance metrics* section in the *IBM Spectrum Scale: Problem Determination Guide*.

measurement
Creates a threshold for the specified measurement. The following measurements are supported:

DataPool_capUtil
Data Pool Capacity Utilization. Calculated as:

sum(gpfs_pool_total_dataKB)-sum(gpfs_pool_free_dataKB)/sum(gpfs_pool_total_dataKB)

DiskIoLatency_read
Average time in milliseconds spent for a read operation on the physical disk. Calculated as:

disk_read_time/disk_read_ios

DiskIoLatency_write
Average time in milliseconds spent for a write operation on the physical disk. Calculated as:

disk_write_time/disk_write_ios

Fileset_inode
Fileset Inode Capacity Utilization. Calculated as:

sum(gpfs_fset_allocInodes)-sum(gpfs_fset_freeInodes)/sum(gpfs_fset_maxInodes)

FsLatency_diskWaitRd
Average disk wait time per read operation on the IBM Spectrum Scale client. Calculated as:

sum(gpfs_fs_tot_disk_wait_rd)/sum(gpfs_fs_read_ops)

FsLatency_diskWaitWr
Average disk wait time per write operation on the IBM Spectrum Scale client. Calculated as:

sum(gpfs_fs_tot_disk_wait_wr)/sum(gpfs_fs_write_ops)

MetaDataPool_capUtil
MetaData Pool Capacity Utilization. Calculated as:

sum(gpfs_pool_total_metaKB)-sum(gpfs_pool_free_metaKB))/sum(gpfs_pool_total_metaKB)

NFSNodeLatency_read
Time taken for NFS read operations. Calculated as:

sum(nfs_read_lat)/*sum(nfs_read_ops)*

NFSNodeLatency_write
Time taken for NFS read operations. Calculated as:

sum(nfs_write_lat)/*sum(nfs_write_ops)*

SMBNodeLatency_read
Total amount of time spent for all type of SMB read requests. Calculated as:

avg(op_time)/*avg(op_count)*

SMBNodeLatency_write
Total amount of time spent for all type of SMB write requests. Calculated as:

avg(op_time)/*avg(op_count)*

MemoryAvailable_percent
Estimated available memory percentage. Calculated as:

- For the nodes having less than 40 GB total memory allocation:

  (*mem_memfree*+*mem_buffers*+*mem_cached*)/*mem_memtotal*
- For the nodes having equal to or greater than 40 GB memory allocation:

  (*mem_memfree*+*mem_buffers*+*mem_cached*)/40000000

--errorlevel
Defines the threshold error limit. The threshold error limit can be a percentage or an integer, depending on the metric on which the threshold value is being set.

--warnlevel
Defines the threshold warn limit. The threshold warn limit can be a percentage or an integer, depending on the metric on which the threshold value is being set.

--direction
Defines the direction for the threshold limit. The allowed values are *high* or *low*.

--groupby
Groups the result based on the group key. The following values are allowed for the group key:

- *gpfs_cluster_name*
- *gpfs_disk_name*
- *gpfs_diskpool_name*
- *gpfs_disk_usage_name*
- *gpfs_fset_name*
- *gpfs_fs_name*
- *mountPoint*
- *netdev_name*
- *node*

--filterby
Filters the result based on the filter key. The following values are allowed for the filter key:

- *gpfs_cluster_name*
- *gpfs_disk_name*
- *gpfs_diskpool_name*
- *gpfs_disk_usage_name*
- *gpfs_fset_name*
- *gpfs_fs_name*
- *mountPoint*

- *netdev_name*
- *node*

--sensitivity
Defines the sample interval value in seconds. It is set to 300 by default. If a sensor is configured with a time interval greater than 300 seconds, then the *--sensitivity* is set to the same value as the *sensors period*. The minimum value allowed is 120 seconds. If a sensor is configured with a time interval less than 120 seconds, the *--sensitivity* is set to 120 seconds.

Starting from IBM Spectrum Scale 5.0.4, the user is allowed to specify the `-min` and `-max` suffix for the sensitivity value to evaluate the original outlier data points within the specified sample interval. The outlier observation works only for the sensitivity time interval that is greater than the sensor's time interval.

--hysteresis
Defines the percentage that the observed value must be under (or over) the current threshold level to switch back to the previous state. The default value is 0.0, while the recommended value is 5.0.

--name
Defines the name of the rule. It can be an alphanumeric string with up to 30 characters. If the rule name is not specified, default name is set. The default name is set using the metric name followed by underscore and then a "custom" prefix.

--errormsg
This is a user defined input. The message can be 256 bytes long. It must be added within double quotes (""), else the system throws an error.

--warnmsg
This is a user defined input. The message can be 256 bytes long. It must be added within double quotes (""), else the system throws an error.

Important:

- The mathematical aggregations: AVG, SUM, MAX, MIN, RATE could be used to determine how to merge the metric values in the evaluation source. The aggregation operations are not supported for measurements.
- For each rule the user can configure up to two conditions, *--error* and *--warn*, triggering event state change. At least one level limit setting is required. For example, the threshold add command must have one of the following options:
  - ```
    mmhealth thresholds add { metric[:sum|avg|min|max|rate] | measurement
    --error{threshold error limit} ---direction {high|low}
    ```
  - ```
    mmhealth thresholds add { metric[:sum|avg|min|max|rate] | measurement
    --error{threshold error limit} --warn{threshold warn limit}
    ```
  - ```
    mmhealth thresholds add { metric[:sum|avg|min|max|rate] | measurement
    --error{threshold error limit} --warn{threshold warn limit} ---direction {high|low}
    ```
- The customer can also influence the measuring quantity and precision by specifying sensitivity, group-by, filter-by, hysteresis, or rule name option setting.
- For each condition level the customer can leave an output message text by using the *--errormsg* or *--warnmsg* options, which is integrated into the state change event message. The state change event message is triggered when this condition is exceeded.
- An existing threshold rule is deactivated in the following cases:
  1. If no metric keys exist for the defined threshold rule in the performance monitoring tool metadata.
  2. If the sensor corresponding to the rule is not enabled.

  As soon as the metric keys and metric values are detected for a rule, the state of the rule switches to `active`.
    ```

thresholds delete
Deletes the threshold rules from the system.

ruleName
Deletes a specific threshold rule.

all
Deletes all the threshold rules.

Note: Using the **mmhealth thresholds delete** command to delete a rule accomplishes the following tasks:

- The rule is removed from the thresholds rules specification file and active monitoring process.
- All the current health information created by this particular rule is removed as well.

config interval
Sets the monitoring interval for the whole cluster.

off
The monitoring is turned off for the whole cluster.

low
Monitoring is set for every (`default monitoring time *10`) seconds.

medium
Monitoring is set for every (`default monitoring time *5`) seconds.

default
Monitoring is set for every 15-30 seconds based on the service being monitored

high
Monitoring is set for every (`default monitoring time /2`) seconds.

config monitor

Displays or changes the health monitoring service status of the current node or of the nodes specified by the `-N` option. This command is used to suspend the health monitoring activities on a specific node or a subset of nodes for maintenance activities. A node can either have its monitoring services in the `paused` state or the `not paused` state. When in a `paused` state, the health monitoring services are all disabled, and the output from running the **mmhealth node show** command displays all the services in a disabled state. By default, a node is in a `not paused` state.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

-N
Allows the system to make remote calls to the other nodes in the cluster for:

Node[,Node....]
Specifies the node or list of nodes for which the health status is displayed.

NodeFile
Specifies a file containing a list of node descriptors, one per line, for which the health status is displayed.

NodeClass
Specifies the node class for which the health status is displayed.

pause

Disables the health monitoring services on the current node or specified nodes.

resume

Resumes the health monitoring services on the current node or specified nodes.

config webhook

Manages the Webhook URLs that receive HTTP POST messages whenever events are generated during health monitoring of a node. Each POST message sent to the Webhook URL is a JSON formatted message containing detailed information for one or more health events that have occurred.

list

Lists all of the configured Webhook URLs.

list [-Y]

Lists all enabled and disabled Webhook URLs. The `-Y` option displays the list in a parseable format with a colon (:) used as a field delimiter. Each column is described by a header. The Webhook URL is encoded such that colons are translated to %3A accordingly.

add <URL>

Adds a Webhook URL to the list of defined Webhook URLs that receive the HTTP POST actions when events are generated during health monitoring. A maximum of 5 URLs could be defined. The same Webhook URL could be re-added and is useful in the event when the URL gets disabled because of persistent communication issues.

remove <URL>

Removes a Webhook URL from the list of configured Webhook URLs.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmhealth` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. See the information about the requirements for administering a GPFS system in the *IBM Spectrum Scale: Administration Guide*.

Examples

1. To show the health status of the current node, issue the following command:

```
# mmhealth node show
```

A sample output is as follows:

```
Node name:      test_node
Node status:    HEALTHY
Status Change:  39 min. ago

Component          Status        Status Change    Reasons
------------------------------------------------------------------
GPFS               HEALTHY       39 min. ago       -
NETWORK            HEALTHY       40 min. ago       -
FILESYSTEM         HEALTHY       39 min. ago       -
DISK               HEALTHY       39 min. ago       -
CES                HEALTHY       39 min. ago       -
PERFMON            HEALTHY       40 min. ago       -
THRESHOLD          HEALTHY       40 min. ago       -
```

2. To view the health status of a specific node, issue the following command:

```
# mmhealth node show -N test_node2
```

A sample output is as follows:

```
Node name:      test_node2
Node status:    CHECKING
Status Change:  Now

Component       Status        Status Change    Reasons
------------------------------------------------------------------
GPFS            CHECKING      Now              -
NETWORK         HEALTHY       Now              -
FILESYSTEM      CHECKING      Now              -
DISK            CHECKING      Now              -
CES             CHECKING      Now              -
PERFMON         HEALTHY       Now              -
```

3. To view the health status of all the nodes, issue the following command:

```
# mmhealth node show -N all
```

A sample output is as follows:

```
Node name:    test_node
Node status:  DEGRADED

Component           Status        Status Change     Reasons
-------------------------------------------------------------
GPFS                HEALTHY          Now             -
CES                 FAILED           Now             smbd_down
FileSystem          HEALTHY          Now             -

Node name:            test_node2
Node status:          HEALTHY

Component           Status        Status Change    Reasons
-------------------------------------------------------------
GPFS                HEALTHY       Now              -
CES                 HEALTHY       Now              -
FileSystem          HEALTHY       Now              -
```

4. To view the detailed health status of the component and its sub-component, issue the following command:

```
# mmhealth node show ces
```

A sample output is as follows:

```
Node name:      test_node

Component       Status        Status Change    Reasons
------------------------------------------------------------------
CES             HEALTHY       2 min. ago       -
  AUTH          DISABLED      2 min. ago       -
  AUTH_OBJ      DISABLED      2 min. ago       -
  BLOCK         DISABLED      2 min. ago       -
  CESNETWORK    HEALTHY       2 min. ago       -
  NFS           HEALTHY       2 min. ago       -
  OBJECT        DISABLED      2 min. ago       -
  SMB           HEALTHY       2 min. ago       -
```

5. To view the health status of only unhealthy components, issue the following command:

```
# mmhealth node show --unhealthy
```

A sample output is as follows:

```
Node name:        test_node
Node status:       FAILED
Status Change:  1 min. ago

Component       Status        Status Change    Reasons
-----------------------------------------------------------------
GPFS            FAILED        1 min. ago       gpfs_down, quorum_down
```

```
FILESYSTEM      DEPEND        1 min. ago       unmounted_fs_check
CES             DEPEND        1 min. ago       ces_network_ips_down, nfs_in_grace
```

6. To view the health status of sub-components of a node's component, issue the following command:

```
# mmhealth node show --verbose
```

A sample output is as follows:

```
Node name:      gssio1-hs.gpfs.net
Node status:    HEALTHY

Component                                 Status              Reasons
------------------------------------------------------------------------
GPFS                                      DEGRADED            -
NETWORK                                   HEALTHY             -
  bond0                                   HEALTHY             -
  ib0                                     HEALTHY             -
  ib1                                     HEALTHY             -
FILESYSTEM                                DEGRADED            stale_mount, stale_mount, stale_mount
  Basic1                                  FAILED              stale_mount
  Basic2                                  FAILED              stale_mount
  Custom1                                 HEALTHY             -
  gpfs0                                   FAILED              stale_mount
  gpfs1                                   FAILED              stale_mount
DISK                                      DEGRADED            disk_down
  rg_gssio1_hs_Basic1_data_0              HEALTHY             -
  rg_gssio1_hs_Basic1_system_0            HEALTHY             -
  rg_gssio1_hs_Basic2_data_0              HEALTHY             -
  rg_gssio1_hs_Basic2_system_0            HEALTHY             -
  rg_gssio1_hs_Custom1_data1_0            HEALTHY             -
  rg_gssio1_hs_Custom1_system_0           DEGRADED            disk_down
  rg_gssio1_hs_Data_8M_2p_1_gpfs0         HEALTHY             -
  rg_gssio1_hs_Data_8M_3p_1_gpfs1         HEALTHY             -
  rg_gssio1_hs_MetaData_1M_3W_1_gpfs0     HEALTHY             -
  rg_gssio1_hs_MetaData_1M_4W_1_gpfs1     HEALTHY             -
  rg_gssio2_hs_Basic1_data_0              HEALTHY             -
  rg_gssio2_hs_Basic1_system_0            HEALTHY             -
  rg_gssio2_hs_Basic2_data_0              HEALTHY             -
  rg_gssio2_hs_Basic2_system_0            HEALTHY             -
  rg_gssio2_hs_Custom1_data1_0            HEALTHY             -
  rg_gssio2_hs_Custom1_system_0           HEALTHY             -
  rg_gssio2_hs_Data_8M_2p_1_gpfs0         HEALTHY             -
  rg_gssio2_hs_Data_8M_3p_1_gpfs1         HEALTHY             -
  rg_gssio2_hs_MetaData_1M_3W_1_gpfs0     HEALTHY             -
  rg_gssio2_hs_MetaData_1M_4W_1_gpfs1     HEALTHY             -
NATIVE_RAID                               DEGRADED            gnr_pdisk_replaceable, gnr_rg_failed,
enclosure_needsservice
  ARRAY                                   DEGRADED            -
    rg_gssio2-hs/DA1                      HEALTHY             -
    rg_gssio2-hs/DA2                      HEALTHY             -
    rg_gssio2-hs/NVR                      HEALTHY             -
    rg_gssio2-hs/SSD                      HEALTHY             -
  ENCLOSURE                               DEGRADED            enclosure_needsservice
    SV52122944                            DEGRADED            enclosure_needsservice
    SV53058375                            HEALTHY             -
  PHYSICALDISK                            DEGRADED            gnr_pdisk_replaceable
    rg_gssio2-hs/e1d1s01                  FAILED              gnr_pdisk_replaceable
    rg_gssio2-hs/e1d1s07                  HEALTHY             -
    rg_gssio2-hs/e1d1s08                  HEALTHY             -
    rg_gssio2-hs/e1d1s09                  HEALTHY             -
    rg_gssio2-hs/e1d1s10                  HEALTHY             -
    rg_gssio2-hs/e1d1s11                  HEALTHY             -
    rg_gssio2-hs/e1d1s12                  HEALTHY             -
    rg_gssio2-hs/e1d2s07                  HEALTHY             -
    rg_gssio2-hs/e1d2s08                  HEALTHY             -
    rg_gssio2-hs/e1d2s09                  HEALTHY             -
    rg_gssio2-hs/e1d2s10                  HEALTHY             -
    rg_gssio2-hs/e1d2s11                  HEALTHY             -
    rg_gssio2-hs/e1d2s12                  HEALTHY             -
    rg_gssio2-hs/e1d3s07                  HEALTHY             -
    rg_gssio2-hs/e1d3s08                  HEALTHY             -
    rg_gssio2-hs/e1d3s09                  HEALTHY             -
    rg_gssio2-hs/e1d3s10                  HEALTHY             -
    rg_gssio2-hs/e1d3s11                  HEALTHY             -
    rg_gssio2-hs/e1d3s12                  HEALTHY             -
    rg_gssio2-hs/e1d4s07                  HEALTHY             -
    rg_gssio2-hs/e1d4s08                  HEALTHY             -
    rg_gssio2-hs/e1d4s09                  HEALTHY             -
    rg_gssio2-hs/e1d4s10                  HEALTHY             -
    rg_gssio2-hs/e1d4s11                  HEALTHY             -
    rg_gssio2-hs/e1d4s12                  HEALTHY             -
    rg_gssio2-hs/e1d5s07                  HEALTHY             -
    rg_gssio2-hs/e1d5s08                  HEALTHY             -
    rg_gssio2-hs/e1d5s09                  HEALTHY             -
    rg_gssio2-hs/e1d5s10                  HEALTHY             -
    rg_gssio2-hs/e1d5s11                  HEALTHY             -
    rg_gssio2-hs/e2d1s07                  HEALTHY             -
    rg_gssio2-hs/e2d1s08                  HEALTHY             -
    rg_gssio2-hs/e2d1s09                  HEALTHY             -
    rg_gssio2-hs/e2d1s10                  HEALTHY             -
    rg_gssio2-hs/e2d1s11                  HEALTHY             -
    rg_gssio2-hs/e2d1s12                  HEALTHY             -
    rg_gssio2-hs/e2d2s07                  HEALTHY             -
    rg_gssio2-hs/e2d2s08                  HEALTHY             -
    rg_gssio2-hs/e2d2s09                  HEALTHY             -
    rg_gssio2-hs/e2d2s10                  HEALTHY             -
    rg_gssio2-hs/e2d2s11                  HEALTHY             -
    rg_gssio2-hs/e2d2s12                  HEALTHY             -
    rg_gssio2-hs/e2d3s07                  HEALTHY             -
    rg_gssio2-hs/e2d3s08                  HEALTHY             -
    rg_gssio2-hs/e2d3s09                  HEALTHY             -
    rg_gssio2-hs/e2d3s10                  HEALTHY             -
    rg_gssio2-hs/e2d3s11                  HEALTHY             -
```

```
    rg_gssio2-hs/e2d3s12                    HEALTHY             -
    rg_gssio2-hs/e2d4s07                    HEALTHY             -
    rg_gssio2-hs/e2d4s08                    HEALTHY             -
    rg_gssio2-hs/e2d4s09                    HEALTHY             -
    rg_gssio2-hs/e2d4s10                    HEALTHY             -
    rg_gssio2-hs/e2d4s11                    HEALTHY             -
    rg_gssio2-hs/e2d4s12                    HEALTHY             -
    rg_gssio2-hs/e2d5s07                    HEALTHY             -
    rg_gssio2-hs/e2d5s08                    HEALTHY             -
    rg_gssio2-hs/e2d5s09                    HEALTHY             -
    rg_gssio2-hs/e2d5s10                    HEALTHY             -
    rg_gssio2-hs/e2d5s11                    HEALTHY             -
    rg_gssio2-hs/e2d5s12ssd                 HEALTHY             -
    rg_gssio2-hs/n1s02                      HEALTHY             -
    rg_gssio2-hs/n2s02                      HEALTHY             -
  RECOVERYGROUP                             DEGRADED            gnr_rg_failed
    rg_gssio1-hs                            FAILED              gnr_rg_failed
    rg_gssio2-hs                            HEALTHY             -
  VIRTUALDISK                               DEGRADED            -
    rg_gssio2_hs_Basic1_data_0              HEALTHY             -
    rg_gssio2_hs_Basic1_system_0            HEALTHY             -
    rg_gssio2_hs_Basic2_data_0              HEALTHY             -
    rg_gssio2_hs_Basic2_system_0            HEALTHY             -
    rg_gssio2_hs_Custom1_data1_0            HEALTHY             -
    rg_gssio2_hs_Custom1_system_0           HEALTHY             -
    rg_gssio2_hs_Data_8M_2p_1_gpfs0         HEALTHY             -
    rg_gssio2_hs_Data_8M_3p_1_gpfs1         HEALTHY             -
    rg_gssio2_hs_MetaData_1M_3W_1_gpfs0     HEALTHY             -
    rg_gssio2_hs_MetaData_1M_4W_1_gpfs1     HEALTHY             -
    rg_gssio2_hs_loghome                    HEALTHY             -
    rg_gssio2_hs_logtip                     HEALTHY             -
    rg_gssio2_hs_logtipbackup               HEALTHY             -
PERFMON                                     HEALTHY             -
```

7. To view the eventlog history of the node for the last hour, issue the following command:

```
# mmhealth node eventlog --hour
```

A sample output is as follows:

```
Node name:      test-21.localnet.com
Timestamp                       Event Name                Severity   Details
2016-10-28 06:59:34.045980 CEST       monitor_started           INFO       The IBM Spectrum Scale monitoring service has
been started
2016-10-28 07:01:21.919943 CEST       fs_remount_mount          INFO       The filesystem objfs was mounted internal
2016-10-28 07:01:32.434703 CEST       disk_found                INFO       The disk disk1 was found
2016-10-28 07:01:32.669125 CEST       disk_found                INFO       The disk disk8 was found
2016-10-28 07:01:36.975902 CEST       filesystem_found          INFO       Filesystem objfs was found
2016-10-28 07:01:37.226157 CEST       unmounted_fs_check        WARNING    The filesystem objfs is probably needed, but
not mounted
2016-10-28 07:01:52.1691 CEST         mounted_fs_check          INFO       The filesystem objfs is mounted
2016-10-28 07:01:52.283545 CEST       fs_remount_mount          INFO       The filesystem objfs was mounted normal
2016-10-28 07:02:07.026093 CEST       mounted_fs_check          INFO       The filesystem objfs is mounted
2016-10-28 07:14:58.498854 CEST       ces_network_ips_down      WARNING    No CES relevant NICs detected
2016-10-28 07:15:07.702351 CEST       nodestatechange_info      INFO       A CES node state change: Node 1 add startup
flag
2016-10-28 07:15:37.322997 CEST       nodestatechange_info      INFO       A CES node state change: Node 1 remove startup
flag
2016-10-28 07:15:43.741149 CEST       ces_network_ips_up        INFO       CES-relevant IPs are served by found NICs
2016-10-28 07:15:44.028031 CEST       ces_network_vanished      INFO       CES NIC eth0 has vanished
```

8. To view the eventlog history of the node for the last hour, issue the following command:

```
# mmhealth node eventlog --hour --verbose
```

A sample output is as follows:

```
Node name:      test-21.localnet.com
Timestamp                       Component     Event Name                Event ID Severity   Details
2016-10-28 06:59:34.045980 CEST       gpfs          monitor_started           999726   INFO       The IBM Spectrum Scale monitoring service has
been started
2016-10-28 07:01:21.919943 CEST       filesystem    fs_remount_mount          999306   INFO       The filesystem objfs was mounted internal
2016-10-28 07:01:32.434703 CEST       disk          disk_found                999424   INFO       The disk disk1 was found
2016-10-28 07:01:32.669125 CEST       disk          disk_found                999424   INFO       The disk disk8 was found
2016-10-28 07:01:36.975902 CEST       filesystem    filesystem_found          999299   INFO       Filesystem objfs was found
2016-10-28 07:01:37.226157 CEST       filesystem    unmounted_fs_check        999298   WARNING    The filesystem objfs is probably needed, but not
mounted
2016-10-28 07:01:52.113691 CEST       filesystem    mounted_fs_check          999301   INFO       The filesystem objfs is mounted
2016-10-28 07:01:52.283545 CEST       filesystem    fs_remount_mount          999306   INFO       The filesystem objfs was mounted normal
2016-10-28 07:02:07.026093 CEST       filesystem    mounted_fs_check          999301   INFO       The filesystem objfs is mounted
2016-10-28 07:14:58.498854 CEST       cesnetwork    ces_network_ips_down      999426   WARNING    No CES relevant NICs detected
2016-10-28 07:15:07.702351 CEST       gpfs          nodestatechange_info      999220   INFO       A CES node state change: Node 1 add startup flag
2016-10-28 07:15:37.322997 CEST       gpfs          nodestatechange_info      999220   INFO       A CES node state change: Node 1 remove startup
flag
2016-10-28 07:15:43.741149 CEST       cesnetwork    ces_network_ips_up        999427   INFO       CES-relevant IPs are served by found NICs
2016-10-28 07:15:44.028031 CEST       cesnetwork    ces_network_vanished      999434   INFO       CES NIC eth0 has vanished
```

9. To view the detailed description of an event, issue the **mmhealth event show** command. This is an example for *quorum_down* event:

```
# mmhealth event show quorum_down
```

A sample output is as follows:

```
Event Name:             quorum_down
Event ID:               999289
Description:            Reasons could be network or hardware issues, or a shutdown of the cluster service.
                        The event does not necessarily indicate an issue with the cluster quorum state.
```

```
Cause:                  The local node does not have quorum. The cluster service might not be running.
User Action:            Check if the cluster quorum nodes are running and can be reached over the network.
                        Check local firewall settings
Severity:               ERROR
State:                  DEGRADED
```

10. To view the list of hidden events, issue the following command:

```
# mmhealth event list HIDDEN
```

A sample output is as follows:

```
Event                         scope
-------------------------------------
gpfs_pagepool_small           -
nfsv4_acl_type_wrong          fs1
nfsv4_acl_type_wrong          fs2
```

11. To view the detailed description of the cluster, issue the following command:

```
# mmhealth cluster show
```

A sample output is as follows:

```
Component     Total   Failed   Degraded   Healthy   Other
-----------------------------------------------------------------
NODE             50        1          1        48       -
GPFS             50        1          -        49       -
NETWORK          50        -          -        50       -
FILESYSTEM        3        -          -         3       -
DISK             50        -          -        50       -
CES               5        -          5         -       -
CLOUDGATEWAY      2        -          -         2       -
PERFMON          48        -          5        43       -
THRESHOLD         4        -          -         4       -
```

Note: The cluster must have the minimum release level as 4.2.2.0 or higher to use **mmhealth cluster show** command.
Also, this command does not support Windows operating system.

12. To view more information of the cluster health status, issue the following command:

```
# mmhealth cluster show --verbose
```

A sample output is as follows:

```
Component     Total   Failed   Degraded   Healthy   Other
-----------------------------------------------------------------
NODE             50        1          1        48       -
GPFS             50        1          -        49       -
NETWORK          50        -          -        50       -
FILESYSTEM
  FS1            15        -          -        15       -
  FS2             5        -          -         5       -
  FS3            20        -          -        20       -
DISK             50        -          -        50       -
CES               5        -          5         -       -
  AUTH            5        -          -         -       5
  AUTH_OBJ        5        5          -         -       -
  BLOCK           5        -          -         -       5
  CESNETWORK      5        -          -         5       -
  NFS             5        -          -         5       -
  OBJECT          5        -          -         5       -
  SMB             5        -          -         5       -
CLOUDGATEWAY      2        -          -         2       -
PERFMON          48        -          5        43       -
THRESHOLD         4        -          -         4       -
```

13. To create a new threshold rule, issue the following command:

```
# mmhealth thresholds add MetaDataPool_capUtil --errorlevel 90 --direction high
--groupby gpfs_fs_name,gpfs_diskpool_name
```

A sample output is as follows:

```
New rule 'MetaDataPool_capUtil_custom' is created.
The monitor process is activated
```

14. To view the list of threshold rules defined for the system, issue the following command:

```
# mmhealth thresholds list
```

A sample output is as follows:

```
### Threshold Rules ###
rule_name                   metric                error  warn    direction  filterBy  groupBy            sensitivity
-----------------------------------------------------------------------------------------------------------------------
InodeCapUtil_Rule           Fileset_inode         90.0   80.0    high                 gpfs_cluster_name, 300
                                                                                      gpfs_fs_name,
                                                                                      gpfs_fset_name
MetaDataPool_capUtil_custom  MetaDataPool_capUtil  90     None    high                 gpfs_fs_name,      300

gpfs_diskpool_name
DataCapUtil_Rule            DataPool_capUtil      90.0   80.0    high                 gpfs_cluster_name, 300
                                                                                      gpfs_fs_name,
                                                                                      gpfs_diskpool_name
MemFree_Rule                mem_memfree           50000  100000  low                  node               300
MetaDataCapUtil_Rule        MetaDataPool_capUtil  90.0   80.0    high                 gpfs_cluster_name, 300
                                                                                      gpfs_fs_name,
                                                                                      gpfs_diskpool_name
```

15. To view the detailed health status of file system component, issue the following command:

```
# mmhealth node show filesystem -v
```

A sample output is as follows:

```
Node name:      gpfsgui-12.novalocal

Component           Status        Status Change            Reasons
------------------------------------------------------------------------
FILESYSTEM          DEGRADED      2016-09-29 15:22:48      pool-data_high_error
  fs1               FAILED        2016-09-29 15:22:48      pool-data_high_error
  fs2               HEALTHY       2016-09-29 15:22:33      -
  objfs             HEALTHY       2016-09-29 15:22:33      -


Event                    Parameter   Severity  Active Since           Event Message
-------------------------------------------------------------------------------------------------------------------
--------
pool-data_high_error     fs1          ERROR     2016-09-29 15:22:47    The pool myPool of file system fs1
                                                                       reached a nearly exhausted data
level. 90.0
inode_normal             fs1          INFO      2016-09-29 15:22:47    The inode usage of fileset root
                                                                       in file system fs1 reached a
normal level.
inode_normal             fs2          INFO      2016-09-29 15:22:47    The inode usage of fileset root
                                                                       in file system fs2 reached a
normal level.
inode_normal             objfs        INFO      2016-09-29 15:22:47    The inode usage of fileset root
                                                                       in file system objfs reached a
normal level.
inode_normal             objfs        INFO      2016-09-29 15:22:47    The inode usage of fileset
                                                                       Object_Fileset in file system
objfs reached
                                                                       a normal level.
mounted_fs_check         fs1          INFO      2016-09-29 15:22:33    The filesystem fs1 is mounted
mounted_fs_check         fs2          INFO      2016-09-29 15:22:33    The filesystem fs2 is mounted
mounted_fs_check         objfs        INFO      2016-09-29 15:22:33    The filesystem objfs is mounted
pool-data_normal         fs1          INFO      2016-09-29 15:22:47    The pool system of file system
                                                                       fs1 reached a normal data level.
pool-data_normal         fs2          INFO      2016-09-29 15:22:47    The pool system of file system
                                                                       fs2 reached a normal data level.
pool-data_normal         objfs        INFO      2016-09-29 15:22:47    The pool data of file system
                                                                       objfs reached a normal data level.
pool-data_normal         objfs        INFO      2016-09-29 15:22:47    The pool system of file system
                                                                       objfs reached a normal data level.
pool-metadata_normal     fs1          INFO      2016-09-29 15:22:47    The pool system of file system
                                                                       fs1 reached a normal metadata
level.
pool-metadata_normal     fs1          INFO      2016-09-29 15:22:47    The pool myPool of file system
                                                                       fs1 reached a normal metadata
level.
pool-metadata_normal     fs2          INFO      2016-09-29 15:22:47    The pool system of file system fs2
                                                                       reached a normal metadata level.
pool-metadata_normal     objfs        INFO      2016-09-29 15:22:47    The pool system of file system
                                                                       objfs reached a normal metadata
level.
pool-metadata_normal     objfs        INFO      2016-09-29 15:22:47    The pool data of file system
```

```
                                                                    objfs reached a normal metadata
level.
```

16. To check the monitoring interval, issue the following command:

```
# mmhealth config interval
```

    A sample output is as follows:

```
Monitor interval is DEFAULT.
```

17. To set the monitoring interval to low, issue the following command:

```
# mmhealth config interval LOW
```

    A sample output is as follows:

```
Monitor interval changed to LOW.
```

18. To solve one of the events that is manually solvable, issue the following command:

```
# mmhealth event resolve out_of_memory
```

    A sample output is as follows:

```
Successfully resolved event out_of_memory for entity with event out_of_memory_ok.
```

    To manually solve one of the events with an entity name, run the following command:

```
# mmhealth event resolve fserrallocblock gpfs0
```

    A sample output is as follows:

```
Successfully resolved event fserrallocblock for entity gpfs0 with event fsstruct_fixed.
```

19. To show the health monitoring state of the current node, issue the following command:

```
# mmhealth config monitor
```

    A sample output is as follows:

```
Node test-node-1.fyre.ibm.com health monitoring is not paused
```

20. To show the current health monitoring state of all the nodes in the cluster, issue the following command:

```
# mmhealth config monitor -N all
```

    A sample output is as follows:

```
Node test-node-1.fyre.ibm.com health monitoring is not paused
Node test-node-2.fyre.ibm.com health monitoring is not paused
Node test-node-3.fyre.ibm.com health monitoring is not paused
```

21. To disable or suspend the health monitoring services on the current node, issue the following command:

```
# mmhealth config monitor pause
```

    A sample output is as follows:

```
Health monitoring services have been paused on node:
test-node-1.fyre.ibm.com
```

22. To enable or resume the health monitoring services on the current node, issue the following command:

```
# mmhealth config monitor resume
```

A sample output is as follows:

```
Health monitoring services have resumed on node:
test-node-1.fyre.ibm.com
```

23. To show the current list of configured Webhook URLS, issue the following command (`list` is optional):

```
# mmhealth config webhook [ list ]
```

A sample output is as follows:

```
http://10.21.38.26:8080/webhook/
https://9.160.48.22:8081/webhook
```

24. To show extended details about the configured Webhook URLs, issue the following command:

```
# mmhealth config webhook list -Y
```

A sample output is as follows:

```
mmhealth:webhook:HEADER:version:reserved:reserved:url:uuid:status:
mmhealth:webhook:0:1:::http%3A//10.21.38.26%3A8080/webhook/:ecc1b560-18c0-41ec-
a0f4-1da17e5c9d8e:enabled:
mmhealth:webhook:0:1:::https%3A//9.160.48.22%3A8081/webhook:ec47997f-
c52a-4dc3-98de-6c6f1d425374:enabled:
```

25. To add a Webhook URL, issue the following command, similar to this, substituting the listed URL with your full Webhook URL:

```
# mmhealth config webhook add https://9.160.48.22:8081/webhook
```

A sample output is as follows:

```
Successfully connected to https://9.160.48.22:8081/webhook
Webhook URL https://9.160.48.22:8081/webhook successfully linked to the health event
monitoring system.
Webhook UUID is ec47997f-c52a-4dc3-98de-6c6f1d425374
```

26. To remove a Webhook URL, issue the following command, similar to this, substituting the listed URL with your full Webhook URL:

```
# mmhealth config webhook remove  https://9.160.48.22:8081/webhook
```

A sample output is as follows:

```
Webhook URL https://9.160.48.22:8081/webhook successfully removed from health event
monitoring
```

See also

- “mmchconfig command” on page 170
- “mmlscluster command” on page 492
- “mmobj command” on page 573
- “mmsmb command” on page 709

Location

`/usr/lpp/mmfs/bin`

mmimgbackup command

Performs a backup of a single GPFS file system metadata image.

Synopsis

```
mmimgbackup Device [-g GlobalWorkDirectory]
            [-L n] [-N {Node[,Node...] | NodeFile | NodeClass}]
            [-S SnapshotName] [--image ImageSetName] [--notsm | --tsm]
            [--qos QOSClass] [--tsm-server ServerName] [POLICY-OPTIONS]
```

Availability

Available on all IBM Spectrum Scale editions. Available on AIX and Linux.

Description

The **mmimgbackup** command performs a backup of a single GPFS file system metadata image.

You must run the **mmbackupconfig** command before you run the **mmimgbackup** command. For more information, see the topic *Scale Out Backup and Restore (SOBAR)* in the *IBM Spectrum Scale: Administration Guide*.

Parameters

Device
The device name of the file system whose metadata image is to be backed up. File system names need not be fully-qualified. `fs0` is as acceptable as `/dev/fs0`.

This must be the first parameter.

-g *GlobalWorkDirectory*
The directory to be used for temporary files that need to be shared between the **mmimgbackup** worker nodes and to hold backup images until sent to archive. The default is:

```
mount_point_for_Device/.mmimgbackup
```

-L *n*
Controls the level of information displayed by the **mmimgbackup** command. The default for **mmimgbackup** is 1. Larger values indicate the display of more detailed information. *n* should be one of the following values:

0
Displays only serious errors.

1
Displays some information as the command executes, but not for each file. This is the default.

2
Displays each chosen file and the scheduled action.

3
Displays the same information as 2, plus each candidate file and the applicable rule.

4
Displays the same information as 3, plus each explicitly EXCLUDEd file and the applicable rule.

5
Displays the same information as 4, plus the attributes of candidate and EXCLUDEd files.

6
Displays the same information as 5, plus non-candidate files and their attributes.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies the nodes that will participate in the backup. This command supports all defined node classes. The default is to run only on the node where the **mmimgbackup** command is running or the current value of the `defaultHelperNodes` parameter of the **mmchconfig** command.

Note: If more than one node is specified, ensure all nodes have the same operating system or unexpected results may occur.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

-S *SnapshotName*
Archives the files in the specified global snapshot rather than in the active file system. The snapshot specified cannot be a fileset-level snapshot.

--image *ImageSetName*
Saves the image files using the provided argument as the base set name. Image file names use the following format:

```
ImageSetName_YYYYMMDD_hh.mm.ss_BBB.sbr
```

or

```
ImageSetName_YYYYMMDD_hh.mm.ss.idx
```

where:

ImageSetName
The default *ImageSetName* is `ImageArchive`.

YYYY
A four-digit year.

MM
A two-digit month.

DD
A two-digit day.

hh
A two-digit hour.

mm
A two-digit minute.

ss
A two-digit second.

BBB
A three-digit bucket number.

--notsm | --tsm
Omits (enables) archiving an image fileset to IBM Spectrum Protect through the `dsmc` commands.

--qos *QosClass*
Specifies the Quality of Service for I/O operations (QoS) class to which the instance of the command is assigned. If you do not specify this parameter, the instance of the command is assigned by default to the `maintenance` QoS class. This parameter has no effect unless the QoS service is enabled. For more information, see the topic "mmchqos command" on page 263. Specify one of the following QoS classes:

`maintenance`
This QoS class is typically configured to have a smaller share of file system IOPS. Use this class for I/O-intensive, potentially long-running GPFS commands, so that they contribute less to reducing overall file system performance.

other
This QoS class is typically configured to have a larger share of file system IOPS. Use this class for administration commands that are not I/O-intensive.

For more information, see the topic *Setting the Quality of Service for I/O operations (QoS)* in *IBM Spectrum Scale: Administration Guide*.

--tsm-server ***ServerName***
Specifies the server name to provide to the IBM Spectrum Protect `dsmc` command used to store the image data files in the IBM Spectrum Protect server.

POLICY-OPTIONS
The following `mmapplypolicy` options may also be used with **mmimgbackup**:

-A ***IscanBuckets***
Specifies the number of buckets of inode numbers (number of inode/filelists) to be created and processed by the parallel inode scan. The default is 17. A bucket will typically represent 1,000,000 in-use inodes.

-a ***IscanThreads***
Specifies the number of threads and sort pipelines each node will run during parallel inode scan and policy evaluation. The default is 4.

-D ***yyyy-mm-dd*[@*hh*:*mm*[:*ss*]]**
Specifies the date and (UTC) time to be used by the **mmimgbackup** command when evaluating the policy rules. The default is the current date and time. If only a date is specified, the time will default to 00:00:00.

-M ***name=value***
Indicates a user defined macro specification. There can be more than one `-M` argument. These macro specifications are passed on to the `m4` preprocessor as `-D` specifications.

-n ***DirThreadLevel***
Specifies the number of threads that will be created and dispatched within each **mmimgbackup** process during the directory scan phase. The default is 24.

-s ***LocalWorkDirectory***
Specifies the directory to be used for local temporary storage during command processing. The default directory is `/tmp`.

--sort-buffer-size ***Size***
Specifies the size for the main memory buffer to be used by sort command.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the **mmimgbackup** command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

To create a backup image with an *ImageSetName* of `sobar.cluster.fs9`, for the snapshot `snap1` of file system `fs9` where the image is stored in the file system `/backup_images` on the IBM Spectrum Protect server, issue:

```
mmimgbackup fs9 -S snap1 -g /backup_images --image sobar.cluster.fs9
```

To show that the images are stored on the IBM Spectrum Protect server, issue this command:

```
dsmls /backup_images
```

The system displays information similar to:

```
/backup_images/4063536/mmPolicy.4260220.51A8A6BF:
        1088         1088            0   r      sobar.cluster.fs9_20121129_16.19.55.idx
        4520         4520            0   r      sobar.cluster.fs9_20121129_16.19.55_000.sbr
```

See also

- “mmapplypolicy command” on page 80
- “mmbackupconfig command” on page 111
- “mmimgrestore command” on page 462
- “mmrestoreconfig command” on page 671

Location

`/usr/lpp/mmfs/bin`

mmimgrestore command

Restores a single GPFS file system from a metadata image.

Synopsis

```
mmimgrestore Device ImagePath [-g GlobalWorkDirectory]
             [-L n] [-N {Node[,Node...] | NodeFile | NodeClass}]
             [--image ImageSetName] [--qos QOSClass] [POLICY-OPTIONS]
```

Availability

Available on all IBM Spectrum Scale editions. Available on AIX and Linux.

Description

The `mmimgrestore` command restores a single GPFS file system from a metadata image.

The `mmrestoreconfig` command must be run prior to running the `mmimgrestore` command. For more information, see *Scale Out Backup and Restore (SOBAR)* in *IBM Spectrum Scale: Administration Guide*.

Parameters

Device
The device name of the file system whose metadata image is to be restored. The file system must be empty and mounted read-only. File system names need not be fully-qualified. `fs0` is as acceptable as `/dev/fs0`.

This must be the first parameter.

ImagePath
The fully-qualified path name to an image fileset containing GPFS backup images. The path must be accessible by every node participating in the restore.

-g *GlobalWorkDirectory*
The directory to be used for temporary files that need to be shared between the `mmimgrestore` worker nodes. If not specified, the default working directory will be the *ImagePath* specified.

-L *n*
Controls the level of information displayed by the `mmimgrestore` command. The default for `mmimgrestore` is 1. Larger values indicate the display of more detailed information. *n* should be one of the following values:

0
Displays only serious errors.

1
Displays some information as the command executes, but not for each file. This is the default.

2
Displays each chosen file and the scheduled action.

3
Displays the same information as 2, plus each candidate file and the applicable rule.

4
Displays the same information as 3, plus each explicitly EXCLUDEd file and the applicable rule.

5
Displays the same information as 4, plus the attributes of candidate and EXCLUDEd files.

6
Displays the same information as 5, plus non-candidate files and their attributes.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies the nodes that will participate in the restore. This command supports all defined node classes. The default is to run only on the node where the `mmimgrestore` command is running or the current value of the `defaultHelperNodes` parameter of the `mmchconfig` command.

Note: If more than one node is specified, ensure all nodes have the same operating system or unexpected results may occur.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

--image *ImageSetName*
Specifies the image set name representing the metadata image to be restored. The *ImageSetName* must match the *ImageSetName_YYYYMMDD_hh.mm.ss* that was created during the backup with the `mmimgbackup` command.

--qos *QosClass*
Specifies the Quality of Service for I/O operations (QoS) class to which the instance of the command is assigned. If you do not specify this parameter, the instance of the command is assigned by default to the `maintenance` QoS class. This parameter has no effect unless the QoS service is enabled. For more information, see the topic "mmchqos command" on page 263. Specify one of the following QoS classes:

`maintenance`
This QoS class is typically configured to have a smaller share of file system IOPS. Use this class for I/O-intensive, potentially long-running GPFS commands, so that they contribute less to reducing overall file system performance.

other
This QoS class is typically configured to have a larger share of file system IOPS. Use this class for administration commands that are not I/O-intensive.

For more information, see the topic *Setting the Quality of Service for I/O operations (QoS)* in *IBM Spectrum Scale: Administration Guide*.

POLICY-OPTIONS
The following `mmapplypolicy` options may also be used with `mmimgrestore`:

-m *ThreadLevel*
The number of threads that will be created and dispatched within each image restore process during the policy execution phase of restore. The default is calculated to divide the work of processing all image files being restored evenly among all nodes specified with `-N`. The valid range is 1 to 20.

-s *LocalWorkDirectory*
Specifies the directory to be used for local temporary storage during command processing. The default directory is `/tmp`.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmimgrestore` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

To restore file system `fs9` with data stored in the image with an *ImageSetName* of `sobar.cluster.fs9_20121129_16.19.55` and execute the restore only on AIX nodes, issue:

```
mmimgrestore fs9 /backup_images/406*/mmP* --image sobar.cluster.fs9__20121129_16.19.55 -N
aixnodes
```

See also

- “mmapplypolicy command” on page 80
- “mmbackupconfig command” on page 111
- “mmimgbackup command” on page 458
- “mmrestoreconfig command” on page 671

Location

`/usr/lpp/mmfs/bin`

mmimportfs command

Imports into the cluster one or more file systems that were created in another GPFS cluster.

Synopsis

```
mmimportfs {Device | all} -i ImportfsFile [-S ChangeSpecFile]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmimportfs` command, in conjunction with the `mmexportfs` command, can be used to move into the current GPFS cluster one or more file systems that were created in another GPFS cluster. The `mmimportfs` command extracts all relevant file system and disk information from the *ExportFilesysData* file specified with the `-i` parameter. This file must have been created by the `mmexportfs` command.

When `all` is specified in place of a file system name, any disks that are not associated with a file system will be imported as well.

If the file systems being imported were created on nodes that do not belong to the current GPFS cluster, the `mmimportfs` command assumes that all disks have been properly moved, and are online and available to the appropriate nodes in the current cluster.

Note: If the disks are part of an IBM Spectrum Scale RAID configuration, this explicitly means moving all the disks and respective storage enclosures.

If any node in the cluster, including the node on which you are running the `mmimportfs` command, does not have access to one or more disks, use the `-S` option to assign NSD servers to those disks.

The `mmimportfs` command attempts to preserve any NSD server assignments that were in effect when the file system was exported.

After the `mmimportfs` command completes, use `mmlsnsd` to display the NSD server names that are assigned to each of the disks in the imported file system. Use `mmchnsd` to change the current NSD server assignments as needed.

After the `mmimportfs` command completes, use `mmlsdisk` to display the failure groups to which each disk belongs. Use `mmchdisk` to make adjustments if necessary.

If you are importing file systems into a cluster that already contains GPFS file systems, it is possible to encounter name conflicts. You must resolve such conflicts before the `mmimportfs` command can succeed. You can use the `mmchfs` command to change the device name and mount point of an existing file system. If there are disk name conflicts, use the `mmcrnsd` command to define new disks and specify unique names (rather than let the command generate names). Then replace the conflicting disks using `mmrpldisk` and remove them from the cluster using `mmdelnsd`.

Results

Upon successful completion of the `mmimportfs` command, all configuration information pertaining to the file systems being imported is added to configuration data of the current GPFS cluster.

Parameters

***Device* | all**
: The device name of the file system to be imported. File system names need not be fully-qualified. `fs0` is as acceptable as `/dev/fs0`. Specify `all` to import all GPFS file systems, as well as all disks that do not currently belong to a file system.

If the specified file system device is an IBM Spectrum Scale RAID-based file system, then all affected IBM Spectrum Scale RAID objects will be imported as well. This includes recovery groups, declustered arrays, vdisks, and any other file systems that are based on these objects. For more information about IBM Spectrum Scale RAID, see *IBM Spectrum Scale RAID: Administration*.

This must be the first parameter.

-i *ImportfsFile*
The path name of the file containing the file system information. This file must have previously been created with the `mmexportfs` command.

-S *ChangeSpecFile*
The path name of an optional file containing disk stanzas or recovery group stanzas, or both, specifying the changes that are to be made to the file systems during the import step.

Prior to GPFS 3.5, the disk information was specified in the form of disk descriptors defined as:

```
DiskName:ServerList:
```

For backward compatibility, the `mmimportfs` command will still accept the traditional disk descriptors, but their use is discouraged.

Disk stanzas have the following format:

```
%nsd:
  nsd=NsdName
  servers=ServerList
  usage=DiskUsage
  failureGroup=FailureGroup
  pool=StoragePool
  device=DiskName
thinDiskType={no | nvme | scsi | auto}
```

where:

nsd=*DiskName*
Is the name of a disk from the file system being imported. This clause is mandatory for the `mmimportfs` command.

servers=*ServerList*
Is a comma-separated list of NSD server nodes. You can specify up to eight NSD servers in this list. The defined NSD will preferentially use the first server on the list. If the first server is not available, the NSD will use the next available server on the list.

When specifying server nodes for your NSDs, the output of the `mmlscluster` command lists the host name and IP address combinations recognized by GPFS. The utilization of aliased host names not listed in the `mmlscluster` command output may produce undesired results.

If you do not define a *ServerList*, GPFS assumes that the disk is SAN-attached to all nodes in the cluster. If all nodes in the cluster do not have access to the disk, or if the file system to which the disk belongs is to be accessed by other GPFS clusters, you must specify a *ServerList*.

To remove the NSD server list, do not specify a value for *ServerList* (remove or comment out the `servers=`*ServerList* clause of the NSD stanza).

usage=*DiskUsage*
Specifies the type of data to be stored on the disk. If this clause is specified, the value must match the type of usage already in effect for the disk; `mmimportfs` cannot be used to change this value.

failureGroup=*FailureGroup*
Identifies the failure group to which the disk belongs. If this clause is specified, the value must match the failure group already in effect for the disk; `mmimportfs` cannot be used to change this value.

pool=*StoragePool*
Specifies the storage pool to which the disk is to be assigned. If this clause is specified, the value must match the storage pool already in effect for the disk; `mmimportfs` cannot be used to change this value.

device=*DiskName*
The block device name of the underlying disk device. This clause is ignored by the `mmimportfs` command.

thinDiskType={no | nvme | scsi | auto}
Specifies the space reclaim disk type:

no
The disk does not support space reclaim. This value is the default.

nvme
The disk is a TRIM capable NVMe device that supports the `mmreclaimspace` command.

scsi
The disk is a thin provisioned SCSI disk that supports the `mmreclaimspace` command.

auto
The type of the disk is either `nvme` or `scsi`. IBM Spectrum Scale will try to detect the actual disk type automatically. To avoid problems, you should replace `auto` with the correct disk type, `nvme` or `scsi`, as soon as you can.

Note: The space reclaim auto-detection is enhanced in IBM Spectrum Scale 5.0.5. Use the auto keyword after you upgrade the cluster to IBM Spectrum Scale 5.0.5 or later.

For more information, see the topic *IBM Spectrum Scale with data reduction storage devices* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Recovery group stanzas have the following format:

```
%rg: rgName=RecoveryGroupName
     servers=Primary[,Backup]
```

where:

RecoveryGroupName
Specifies the name of the recovery group being imported.

***Primary*[,*Backup*]**
Specifies the primary server and, optionally, a backup server to be associated with the recovery group.

Notes:

1. You cannot change the name of a disk. You cannot change the disk usage or failure group assignment with the `mmimportfs` command. Use the `mmchdisk` command for this purpose.
2. All disks that do not have stanzas in *ChangeSpecFile* are assigned the NSD servers that they had at the time the file system was exported. All disks with NSD servers that are not valid are assumed to be SAN-attached to all nodes in the cluster. Use the `mmchnsd` command to assign new or change existing NSD server nodes.
3. Use the `mmchrecoverygroup` command to activate recovery groups that do not have stanzas in *ChangeSpecFile*. The `mmchrecoverygroup` command is documented in *IBM Spectrum Scale RAID: Administration*.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmimportfs` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

To import all file systems in the current cluster, issue the following command:

```
# mmimportfs all -i /u/admin/exportfile
```

A sample output is as follows:

```
mmimportfs: Processing file system fs1 ...
mmimportfs: Processing disk gpfs2nsd
mmimportfs: Processing disk gpfs3nsd
mmimportfs: Processing disk gpfs4nsd

mmimportfs: Processing file system fs2 ...
mmimportfs: Processing disk gpfs1nsd1
mmimportfs: Processing disk gpfs5nsd

mmimportfs: Processing disks that do not belong to any file system ...
mmimportfs: Processing disk gpfs6nsd
mmimportfs: Processing disk gpfs1001nsd

mmimportfs: Committing the changes ...

mmimportfs: The following file systems were successfully imported:
        fs1
        fs2
mmimportfs: 6027-1371 Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

See also

- “mmexportfs command” on page 408

Location

`/usr/lpp/mmfs/bin`

mmkeyserv command

Manages encryption key servers and clients.

Synopsis

```
mmkeyserv server {add | update}
   ServerName [--port RestPortNumber] [--user-id RestUserID]
   [--server-pwd PasswordFile] [--accept] [--kmip-cert CertFilesPrefix]
   [--backup ServerName[,ServerName...]] [--distribute | --nodistribute]
   [--timeout ConnectionTimeout] [--retry ConnectionAttempts]
   [--interval Microseconds]
```

or

```
mmkeyserv server delete ServerName
```

or

```
mmkeyserv server show [ServerName] [-Y]
```

or

```
mmkeyserv tenant add TenantName
    --server ServerName [--server-pwd PasswordFile]
```

or

```
mmkeyserv tenant delete TenantName
   --server ServerName [--server-pwd PasswordFile]
```

or

```
mmkeyserv tenant show [TenantName] [--server ServerName] [-Y]
```

or

```
mmkeyserv key create --server ServerName [--server-pwd PasswordFile]
   --tenant TenantName [--count NumberOfKeys]
```

or

```
mmkeyserv key delete --server ServerName [--server-pwd PasswordFile]
   {--all --tenant TenantName | --file ListOfKeysFile}
```

or

```
mmkeyserv key show --server ServerName [--server-pwd PasswordFile]
   --tenant TenantName
```

or

```
mmkeyserv client create ClientName
   --server ServerName [--server-pwd PasswordFile]
   [--cert ClientCertFile --priv ClientPrivateKeyFile
    {--ca-cert CACertFilePrefix | --ca-chain CACertChainFile} ]
   [--days DaysToExpiration][--keystore-pwd PasswordFile]
```

or

```
mmkeyserv client delete ClientName
```

or

```
mmkeyserv client register ClientName
   --rkm-id RkmID --tenant TenantName [--server-pwd PasswordFile]
```

or

```
mmkeyserv client deregister ClientName
   --tenant TenantName [--server-pwd PasswordFile]
```

or

```
mmkeyserv client show [ClientName | --server ServerName] [-Y]
```

or

```
mmkeyserv client update ClientName [--client NewClientName]
    [--cert ClientCertFile --priv ClientPrivateKeyFile
     {--ca-cert CACertFilePrefix | --ca-chain CACertChainFile} ]
    [--force] [--days DaysToExpiration] [--keystore-pwd PasswordFile]
    [--server-pwd PasswordFile]
```

or

```
mmkeyserv rkm change RkmID {[--rkm-id NewRkmID]
   [--backup ServerName[,ServerName...]] [--distribute | --nodistribute]
   [--timeout ConnectionTimeout] [--retry ConnectionAttempts]
   [--interval Microseconds]}
```

or

```
mmkeyserv rkm show
```

Availability

Available with IBM Spectrum Scale Advanced Edition, IBM Spectrum Scale Data Management Edition, IBM Spectrum Scale Developer Edition, or IBM Spectrum Scale Erasure Code Edition.

Description

With the `mmkeyserv` command, you can configure a cluster and a remote key manager (RKM) server so that nodes in the cluster can retrieve master encryption keys when they need to. You must set up an RKM server before you run this command. The RKM server software must be IBM Security Key Lifecycle Manager (SKLM). Nodes in the cluster must have direct network access to the RKM server.

With this command you can connect to an RKM server, create GPFS tenants, create encryption keys, and create and register key clients. The command automatically generates and exchanges certificates and sets up a local keystore. You can also use this command to securely delete key clients, encryption keys, and tenants. You can run this command from any node in the cluster. Each node has a configuration file and a copy of the local keystore. Configuration changes affect all nodes in the cluster.

Password files: Several of the command options require a password file as a parameter. A password file is a text file that contains a password at the beginning. A password must be 1 - 20 characters in length. Only the following characters are allowed in the password:

```
0 to 9
A to Z
a to z
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

Because the password file is a security-sensitive file, it must have the following characteristics:

- It must be a regular file.
- It must be owned by the root user.
- Only the root user must have permission to read or write it.

The following terms are used:

client or key client
An entity in the cluster that represents the nodes that access encrypted files. The key client receives master encryption keys from the tenant of the RKM server.

IBM Security Key Lifecycle Manager (SKLM)
Required key management server software.

Master encryption key (MEK)
A key for encrypting file encryption keys.

Remote key management (RKM) server
A server with software that authenticates clients and provides them with master encryption keys.

RKM.conf file
A configuration file that the mmkeyserv command maintains. Each node in the cluster has a copy of this file. The full path is `/var/mmfs/ssl/keyServ/RKM.conf`.

RKM stanza
A block of configuration information that describes a registered key client. RKM stanzas are stored in the `RKM.conf` file.

RKM ID
An identifier for an RKM stanza.

tenant
A device group in SKLM that contains MEKs for registered key clients.

Parameters

server
Manages a connection with an RKM server.

add
Adds an RKM server connection to the IBM Spectrum Scale cluster. You can adjust the values of some of these options later with the `mmkeyserv rkm change` command.

ServerName
Specifies the host name or IP address of the RKM server.

--port ***RestPortNumber***
Specifies the port number for the Representational State Transfer (REST) interface on the SKLM server:

- If SKLM is configured to use its default REST port for communications with its clients, you do not need to specify this parameter. IBM Spectrum Scale automatically tries to connect with SKLM through the default REST port number of each of the supported versions of SKLM serially, starting with the earliest supported version. If IBM Spectrum Scale successfully connects with SKLM through the default REST port and successfully retrieves the REST certificates, it stops searching for a port and uses the successful port number for future communications with SKLM.

  Note: The default SKLM REST port number depends on the version of SKLM that is installed on the RKM server. For more information, see *Firewall recommendations for IBM SKLM* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

- If SKLM is not configured to use its default REST port number, you must specify the `--port` parameter with the correct port number so that IBM Spectrum Scale can connect with SKLM. If you do not specify a port number or if you specify the incorrect port number, IBM Spectrum Scale fails to connect with SKLM and displays an error message.

For more information, see Part 2 of *Simplified setup: Using SKLM with a self-signed certificate* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

--user-id ***RestUserID***
Specifies the user ID for the RKM server. The default value is `SKLMAdmin`.

--server-pwd *PasswordFile*
Specifies a password file that contains a password for accessing the RKM server. See the requirements for password files in the Description section of this topic. If this option is omitted, the command prompts for a password.

--accept
Configures the command to automatically accept certificates from the RKM server. The acceptance prompt is suppressed.

[--kmip-cert *CertFilesPrefix***]**
Specifies the path and the file name prefix of non-self-signed certificate files in a certificate chain.

Important:

- You must use this option when the specified key server is using a chain of certificates from a certificate authority (CA) or other non-self-signed certificate chain for communication on the KMIP port.
- If you omit this option, the command assumes that the specified key server is using a self-signed certificate for communication on the KMIP port. The command retrieves the self-signed certificate from the key server automatically.

For more information about adding certificates of either kind to the configuration, see the topic *Configuring encryption with SKLM: Simplified setup* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

The certificate files must be formatted as PEM-encoded X.509 certificates. You must manually retrieve the certificate files from the key server. Copy the files to the node on which you are issuing the `mmkeyserv` command. Rename the files so that the full path and the file name of each file in the chain has the following format:

```
CertFilesPrefix.n.cert
```

where:

CertFilesPrefix
Is the full path and the file name prefix of the certificate file.

n
Is an integer that identifies the place of the certificate in the certificate chain:

`0` indicates that the file is the root CA certificate.
An integer in the range `1 - (n-1)` indicates that the file is an intermediate CA certificate.
`n` indicates that the file is the endpoint certificate.

Note: A valid certificate chain can contain zero or more intermediate certificates.

cert
Is the suffix of the certificate file.

For example, in the following set of certificate files, the *CertFilesPrefix* is `/tmp/certificate/sklmChain`:

`/tmp/certificate/sklmChain.0.cert` contains the root certificate.
`/tmp/certificate/sklmChain.1.cert` contains an intermediate certificate.
`/tmp/certificate/sklmChain.2.cert` contains the endpoint certificate.

--backup *ServerName***[,***ServerName...***]**
Specifies a comma-separated list of server names that you want to add to the list of backup RKM servers in the `RKM.conf` file. If an IBM Spectrum Scale node cannot retrieve a master encryption key from its main RKM server, it tries each backup server in the list until it either retrieves a key or exhausts the list.

Note: The `mmkeyserv` command itself does not attempt to contact backup servers or to replicate client information across background servers. The system administrator is responsible for maintaining replication across backup servers.

--distribute | **--nodistribute**

--distribute
Attempts to arrange the list of RKM server names (main RKM server and backup RKM servers) in the `RKM.conf` file in a different order on each node so that each node connects with the servers in a different order. This option provides some performance advantage in retrieving MEKs. This option is the default.

--nodistribute
Does not attempt to arrange the list of backup RKM server names in the `RKM.conf` file.

--timeout *ConnectionTimeout*
Sets the connection timeout in seconds for retrieving an MEK from an RKM server. The valid range is 1 - 120 seconds. The default value is 60 seconds.

--retry *ConnectionAttempts*
Sets the number of attempts to retry a connection to an RKM server. The valid range is 1 - 10 retries. The default value is three retries.

--interval *Microseconds*
Specifies the number of microseconds to wait between connection retries. The valid range is 1 - 1000000000. The default value is 10000 (0.1 seconds).

update
Updates a connection between an IBM Spectrum Scale cluster and an RKM server.

- The command always gets a fresh server certificate from the RKM server.
- If you do not specify the `--port` option, the command first tries to connect with SKLM through the most recently used REST interface port number. If this connection fails, the command tries to connect with SKLM through the default REST interface port number of each of the supported versions of SKLM serially, starting with the earliest supported version. For more information, see the description of the `--port` option for the `mmkeyserv server add` command earlier in this topic.

ServerName
Specifies the host name or IP address of the RKM server.

--port *RestPortNumber*
Specifies the port number for the Representational State Transfer (REST) interface.

--user-id *RestUserID*
Specifies the user ID for the RKM server. The default value is `SKLMAdmin`.

--server-pwd *PasswordFile*
Specifies a password file that contains a password for accessing the RKM server. See the requirements for password files in the Description section of this topic. If this option is omitted, the command prompts for a password if one is required.

--accept
Configures the command to automatically accept certificates from the RKM server. The acceptance prompt is suppressed.

[--kmip-cert *CertFilesPrefix***]**
Specifies the path and the file name prefix of non-self-signed certificate files in a certificate chain.

Important:

- You must use this option when the specified key server is using a chain of certificates from a certificate authority or other non-self-signed certificate chain for communication on the KMIP port.

- If you omit this option, the command assumes that the specified key server is using a self-signed certificate for communication on the KMIP port. The command retrieves the self-signed certificate from the key server automatically.

For more information about adding certificates of either kind to the configuration, see the topic *Configuring encryption with SKLM: Simplified setup* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

The certificate files must be formatted as PEM-encoded X.509 certificates. You must manually retrieve the certificate files from the key server. Copy the files to the node on which you are issuing the `mmkeyserv` command. Rename the files so that the full path and the file name of each file in the chain has the following format:

```
CertFilesPrefix.n.cert
```

where:

CertFilesPrefix
Is the full path and the file name prefix of the certificate file.

n
Is an integer that identifies the place of the certificate in the certificate chain:

`0` indicates that the file is the root CA certificate.
An integer in the range `1 - (n-1)` indicates that the file is an intermediate CA certificate.
n indicates that the file is the endpoint certificate.

Note: A valid certificate chain can contain zero or more intermediate certificates.

cert
Is the suffix of the certificate file.

For example, in the following set of certificate files, the *CertFilesPrefix* is `/tmp/certificate/sklmChain`:

`/tmp/certificate/sklmChain.0.cert` contains the root certificate.
`/tmp/certificate/sklmChain.1.cert` contains an intermediate certificate.
`/tmp/certificate/sklmChain.2.cert` contains the endpoint certificate.

--backup *ServerName*[,*ServerName...*]
Specifies a comma-separated list of server names that you want to add to the backup RKM servers that are listed in the `RKM.conf` file. If an IBM Spectrum Scale node cannot retrieve a master encryption key from its main RKM server, it tries each backup server in the list until it either retrieves a key or exhausts the list.

Important: To remove a backup list, specify **delete** instead of a list of server names, as in the following example:

```
mmkeyserv server update ServerName --backup delete
```

The **backup** option does not change the `RKM.conf` file.

Note: The `mmkeyserv` command itself does not attempt to contact backup servers or to replicate client information across background servers. The system administrator is responsible for maintaining replication across backup servers.

--distribute | --nodistribute
You must specify one of these two options. There is no default value.

--distribute
Attempts to arrange the RKM server names (main RKM server and backup RKM servers) that are listed in the `RKM.conf` file in a different order on each node so that each node connects with the servers in a different order. This option provides some performance advantage in retrieving MEKs.

--nodistribute
Does not attempt to arrange the backup RKM server names that are listed in the RKM.conf file.

Neither option changes the RKM.conf file.

--timeout ***ConnectionTimeout***
Sets the connection timeout in seconds for retrieving an MEK from an RKM server. The valid range is 1 - 120 seconds. To restore the system default value (60 seconds), you can specify either 60 or the keyword **default** as the *ConnectionTimeout* value.

This option does not change the RKM.conf file.

--retry ***ConnectionAttempts***
Sets the number of attempts to retry a connection to an RKM server. The valid range is 1 - 10 retries. To restore the system default value (3 retries), you can specify either 3 or the keyword **default** as the *ConnectionAttempts* value.

This option does not change the RKM.conf file.

--interval ***Microseconds***
Specifies the number of microseconds to wait between connection retries. The valid range is 1 - 1000000000. To restore the system default value (10000, which is 0.1 seconds), you can specify either 10000 or the keyword **default** as the *Microseconds* value.

This option does not change the RKM.conf file.

delete
Removes a connection with an RKM server from the cluster.

ServerName
Specifies the host name or IP address of the RKM server that you want to disconnect from.

show
Displays information about RKM servers. The following table shows the display options:

Table 23. mmkeyserv server show

Option	Displays information about
show	All servers
show *ServerName*	The specified server

ServerName
Specifies the host name or IP address of an RKM server.

tenant
Manages tenants on RKM servers. A tenant is an SKLM device group for holding encryption keys.

add
Specifies the name of a tenant to add to the IBM Spectrum Scale cluster.

- If the tenant is already added to the cluster, the command returns with an error.
- If the tenant exists on the RKM server but is not added to the cluster, the command adds the tenant to the cluster.
- If the tenant does not exist on the RKM server, the command creates the tenant on the server and adds the tenant to the cluster.

TenantName
Specifies the name of the tenant that you want to create.

--server ***ServerName***
Specifies the name of the RKM server to which the tenant belongs.

--server-pwd ***PasswordFile***
Specifies a password file that contains a password for accessing the RKM server. See the requirements for password files in the Description section of this topic. If this option is omitted, the command prompts for a password.

delete
Deletes a tenant from an RKM server.

Note:

- If you delete a tenant that has encryption keys on the key server, the command deletes the tenant from the cluster configuration but not from the key server.
- If you delete a tenant that has no encryption keys on the key server, the command deletes the tenant from both the cluster configuration and the key server.

TenantName
Specifies the name of the tenant that you want to delete.

--server ***ServerName***
Specifies the name of the RKM server to which the tenant belongs.

--server-pwd ***PasswordFile***
Specifies a password file that contains a password for accessing the RKM server. See the requirements for password files in the Description section of this topic. If this option is omitted, the command prompts for a password.

show
Displays information about tenants and RKM servers. The following table shows the results of various combinations of options:

Table 24. mmkeyserv tenant show

Option	Displays information about
`show`	All tenants from all RKM servers
`show` *TenantName*	The specified tenant
`show --server` *ServerName*	All tenants from the specified RKM server
`show` *TenantName* `--server` *ServerName*	The specified tenant and the specified server

TenantName
Specifies the name of a tenant.

--server ***ServerName***
Specifies the name of an RKM server.

key
Manages encryption keys.

create
Creates encryption keys in a tenant and displays the key IDs on the console.

Note: Make a note of the key IDs. You must specify an encryption key ID and an RKM ID when you write an encryption policy rule.

--server ***ServerName***
Specifies the host name or IP address of an RKM server.

--server-pwd ***PasswordFile***
Specifies a password file that contains a password for accessing the RKM server. See the requirements for password files in the Description section of this topic. If this option is omitted, the command prompts for a password.

--tenant ***TenantName***
Specifies the name of the tenant in which you want to create the encryption keys.

--count *NumberOfKeys*
Specifies the number of keys to create. The default value is 1.

delete
Deletes encryption keys from a tenant.

CAUTION: When you delete an encryption key, any data that was encrypted by that key becomes unrecoverable.

--server *ServerName*
Specifies the host name or IP address of an RKM server.

--server-pwd *PasswordFile*
Specifies a password file that contains a password for accessing the RKM server. See the requirements for password files in the Description section of this topic. If this option is omitted, the command prompts for a password.

--all --tenant *TenantName*
Deletes all the encryption keys in the specified tenant.

--file *ListOfKeysFile*
Specifies a file that contains a list of the key IDs of encryption keys that you want to delete, one key per line.

show
Displays information about the encryption keys in a tenant.

--server *ServerName*
Specifies the host name or IP address of an RKM server.

--server-pwd *PasswordFile*
Specifies a password file that contains a password for accessing the RKM server. See the requirements for password files in the Description section of this topic. If this option is omitted, the command prompts for a password.

--tenant *TenantName*
Specifies the name of the tenant that contains the keys that you want to display.

client
Manages key clients. The following facts are important:

- You need only one key client per cluster per RKM server. However, you can create and use multiple key clients on the same RKM server.
- You can register only one key client per tenant per cluster. However, you can register one key client to more than one tenant in the same RKM server.

create
Creates a key client to communicate with the RKM server.

ClientName
Specifies the name of the key client that you want to create. A key client name must be 1 - 16 characters in length and must be unique within an IBM Spectrum Scale cluster.

--server *ServerName*
Specifies the name of the RKM server to which the key client belongs.

--server-pwd *PasswordFile*
Specifies a password file that contains a password for accessing the RKM server. See the requirements for password files in the Description section of this topic. If this option is omitted, the command prompts for a password.

[--cert *ClientCertFile* **--priv** *ClientPrivateKeyFile*
{--ca-cert *CACertFilePrefix* **| --ca-chain** *CACertChainFile***}]**

--cert *ClientCertFile*
Specifies a file that contains a client certificate from a CA.

--priv *ClientPrivateKeyFile*
Specifies a file that contains a client private key that matches the client certificate.

--ca-cert *CACertFilePrefix*
Specifies the path and file name prefix of the certificates that signed the client certificate.

--ca-chain *CACertChainFile*
Specifies a file that contains the certificates of the CA that signed the client certificate.

Important:

- If you are providing a client certificate from a CA for communicating with the remote key server on the KMIP port, you must specify either the `--ca-cert` option or the `--ca-chain` option.
- If you omit the `--cert` option, the command generates a self-signed client certificate for communicating with the remote key server on the KMIP port.

For more information, see the following topics:

Simplified setup: Using SKLM with a self-signed certificate in the *IBM Spectrum Scale: Administration Guide*
Simplified setup: Using SKLM with a certificate chain in the *IBM Spectrum Scale: Administration Guide*

The following requirements must be met:

- The contents of a private key file must be PEM-encoded and unencrypted.
- The CA certificate chain can be specified either as a certificate chain file that contains all the CA certificates or as a set of files, one file for each CA certificate in the chain.
- If a certificate chain file is used, the certificates in it must be in PEM-encoded x509 format and must be concatenated. The CA root certificate must be first, followed by the intermediate CA certificates in order, followed by the final CA certificate that signed the client certificate.
- If certificate files are used, one file for each certificate, the certificates must be in PEM-encoded x509 format. Each file must be renamed in the format *<CACertFilesPrefix><n>*`.cert`, where *<CACertFilesPrefix>* is the full path prefix for the CA certificate files, such as `/tmp/CA/certfiles`, and *<n>* is a CA certificate index. The index is 0 for the CA root certificate and *n* - 1 for the last intermediate CA certificate that signed the client certificate. In the following example, the chain consists of a CA root certificate file and two intermediate CA certificate files. The full path prefix is `/tmp/CA/certfiles`:

CA root certificate	`/tmp/CA/certfiles.0.cert`
First intermediate CA root certificate:	`/tmp/CA/certfiles.1.cert`
Second intermediate CA root certificate:	`/tmp/CA/certfiles.2.cert`

--days *DaysToExpiration*
Specifies the number of days until the newly created client certificate expires. The valid range is 1 - 18262. The default value is 1095. This parameter is not available when you specify a CA-signed certificate chain for the client certificate. The certificates in the CA certificate chain specify their expiration dates.

--keystore-pwd *PasswordFile*
Specifies a password file that contains a client keystore password. See the requirements for password files in the Description section of this topic. If this parameter is omitted the command prompts for a keystore password.

delete
Deletes a key client.

ClientName
Specifies the name of the key client that you want to delete.

register
Registers a key client to a tenant.

ClientName
Specifies the name of the key client that you want to register.

--rkm-id ***RkmID***
Specifies a new RKM ID. An RKM ID must be unique within the cluster, must be 1 - 21 characters in length, and can contain only alphanumeric characters or underscore (_). It must begin with a letter or an underscore. An RKM ID identifies an RKM stanza in the `RKM.conf` file. The stanza contains the information that a node needs to retrieve a master encryption key (MEK) from an RKM.

--tenant ***TenantName***
Specifies the name of the tenant to which you want to register the client.

--server-pwd ***PasswordFile***
Specifies a password file that contains a password for accessing the RKM server. See the requirements for password files in the Description section of this topic. If this option is omitted, the command prompts for a password.

deregister
Unregisters a key client from a tenant.

ClientName
Specifies the name of the key client that you want to unregister.

--tenant ***TenantName***
Specifies the name of the tenant that you want to unregister the key client from.

--server-pwd ***PasswordFile***
Specifies a password file that contains a password for accessing the RKM server. See the requirements for password files in the Description section of this topic. If this option is omitted, the command prompts for a password.

show
Displays information about a key client.

ClientName
Specifies the name of the client whose information you want to display.

--server ***ServerName***
Specifies the name of the server to which the client belongs.

update
Replaces an expired or unexpired client certificate with a new one in the specified key client.

ClientName
Specifies the name of the key client that you want to update.

--client ***NewClientName***
Specifies a new name for client. A key client name must be 1 - 16 characters in length and must be unique within an IBM Spectrum Scale cluster. If this option is omitted, the client name is not changed.

[--cert ***ClientCertFile*** **--priv** ***ClientPrivateKeyFile***
{--ca-cert ***CACertFilePrefix*** **| --ca-chain** ***CACertChainFile*****}]**

--cert ***ClientCertFile***
Specifies a file that contains a client certificate from a CA.

--priv ***ClientPrivateKeyFile***
Specifies a file that contains a client private key that matches the client certificate.

--ca-cert ***CACertFilePrefix***
Specifies the path and file name prefix of the certificates that signed the client certificate.

--ca-chain *CACertChainFile*
Specifies a file that contains the certificates of the CA that signed the client certificate.

For more information, see the descriptions of these parameters in the description of the `mmkeyserv client create` command earlier in this topic.

--force
Generates a self-signed client certificate for the key client. This option is required only if both the following conditions are true:

- You want the key client to have a self-signed client certificate.
- The key client was created with or was previously updated with a client CA-signed certificate and chain.

This option is not required in either of the following situations:

- You want to replace a self-signed certificate with a CA-signed certificate and chain.
- You want to replace a CA-signed certificate and chain with another CA-signed certificate and chain.

--days *DaysToExpiration*
Specifies the number of days until the new client certificate expires. The valid range is 1 - 18262. If this option is omitted, the new client certificate expires in 1095 days. This parameter is not available when you specify a CA-signed certificate chain for the client certificate. The certificates in the CA certificate chain specify their expiration dates.

--keystore-pwd *PasswordFile*
Specifies a password file that contains a client keystore password. See the requirements for password files in the Description section of this topic. If this option is omitted, the current password is not changed.

--server-pwd *PasswordFile*
Specifies a password file that contains a password for accessing the RKM server. See the requirements for password files in the Description section of this topic. If this option is omitted, the command prompts for a password.

rkm

change
Changes the properties of an RKM stanza. For more information about an RKM stanza, see the Description section of this topic.

RkmID
Specifies the RKM ID of the stanza whose properties you want to change.

--rkm-id *RkmID*
Specifies a new RKM ID. An RKM ID must be unique within the cluster, must be 1 - 21 characters in length, and can contain only alphanumeric characters or underscore (_). It must begin with a letter or an underscore. An RKM ID identifies an RKM stanza in the `RKM.conf` file. The stanza contains the information that a node needs to retrieve a master encryption key (MEK) from an RKM.

--backup *ServerName*[,*ServerName...*]
Specifies a comma-separated list of server names that you want to add to the backup RKM servers that are listed in the `RKM.conf` file. If an IBM Spectrum Scale node cannot retrieve a master encryption key from its main RKM server, it tries each backup server in the list until it either retrieves a key or exhausts the list.

Important: To remove a backup list, specify **delete** instead of a list of server names, as in the following example:

```
mmkeyserv rkm change RkmID --backup delete
```

The **backup** option does not change the `RKM.conf` file.

Note: The mmkeyserv command itself does not attempt to contact backup servers or to replicate client information across background servers. The system administrator is responsible for maintaining replication across backup servers.

--distribute | --nodistribute

--distribute
Attempts to arrange the list of RKM server names (main RKM server and backup RKM servers) in the RKM.conf file in a different order on each node so that each node connects with the servers in a different order. This option provides some performance advantage in retrieving MEKs.

--nodistribute
Does not attempt to arrange the list of backup RKM server names in the RKM.conf file.

--timeout ***ConnectionTimeout***
Sets the connection timeout in seconds for retrieving an MEK from an RKM server. The valid range is 1 - 120 seconds. To restore the system default value (60 seconds), you can specify either 60 or the keyword **default** as the *ConnectionTimeout* value.

This option does not change the RKM.conf file.

--retry ***ConnectionAttempts***
Sets the number of attempts to retry a connection to an RKM server. The valid range is 1 - 10 retries. To restore the system default value (3 retries), you can specify either 3 or the keyword **default** as the *ConnectionAttempts* value.

This option does not change the RKM.conf file.

--interval ***Microseconds***
Specifies the number of microseconds to wait between connection retries. The valid range is 1 - 1000000000. To restore the system default value (10000, which is 0.1 seconds), you can specify either 10000 or the keyword **default** as the *Microseconds* value.

This option does not change the RKM.conf file.

show
Displays information about all the RKM stanzas in the RKM.conf file of the node.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

Exit status

0
Successful completion.

Nonzero
A failure occurred.

Security

You must have root authority to run the mmkeyserv command.

The node on which you enter the command must be able to execute remote shell commands on any other administration node in the cluster. It must be able to do so without the use of a password and without producing any extraneous messages. For more information, see the topic *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

Examples

Examples 1 - 5 illustrate the steps in configuring an RKM server and a key client and generating an encryption key:

1. The following command makes an RKM server known to an IBM Spectrum Scale cluster. The name `keyserver01` is the host name of the SKLM server:

```
# mmkeyserv server add keyserver01
Enter password for the RKM server keyserver01:
The security certificate(s) from keyserver01.gpfs.net must be accepted to continue.  View
the
certificate(s) to determine whether you want to trust the certifying authority.
Do you want to view or trust the certificate(s)? (view/yes/no) view

Serial number:          01022a8adf20f3
SHA-256 digest:         2ca4a48a3038f37d430162be8827d91eb584e98f5b3809047ef4a1c72e15fc4c
Signature:
7f0312e7be18efd72c9d8f37dbb832724859ba4bb5827c230e2161473e0753b367ed49d9935
05bd23858541475de8e021e0930725abbd3d25b71edc8fc3de20b7c2db5cd4e865f41c7c410c1d710acf222e1c451
89108e
40568ddcbeb21094264da60a1d96711015a7951eb2655363309d790ab44ee7b26adf8385e2c210b8268c5aede5f82
f26855
4a6fc22ece6efeee2a6264706e71416a0dbe8c39ceacd86054d7cc34dda4fffea4605c037d32129055610821af85d
d9819a
4d7e4baa70c51addcda720d33bc9f8bbde6d292c028b2f525a0275ebea968c26f8f0c4b604719ae3b04e71ed7a818
8cd6ad
f68764374b29c91df3d101a941bf8b7189485ad72
Signature algorithm:    SHA256WithRSASignature
Key size:               2048
Issuer:                 C=US, O=IBM, OU=SKLMNode, SKLMCell, Root Certificate,
CN=c40bbc1xn3.gpfs.net
Subject:                C=US, O=IBM, OU=SKLMNode, SKLMCell, CN=c40bbc1xn3.gpfs.net

Serial number:          01022a24475466
SHA-256 digest:         077c3b53c5046aa893b760c11cca3a993efbc729479771e03791f9ed4f716879
Signature:
227b5befe89f2e55ef628da6b50db1ab842095a54e1505655e3d95fee753a7f7554868aa79b
294c503dc34562cf69c2a2012879675883896856 5c0812c4aedbb0543d396646a269c02bf4c5ce5acba4409a10eff
bd47ca
38ce492698e2dcdc8390b9ae3f4a47c23ee3045ff0145218668f35a63edac68201789ed0db6e5c170f5c6db49769f
0b4c9a
5f208746e4342294c447793ed087fa0ac762588faf420febeb3fca411e4e725bd46476e1f9f44759a696573af5dbb
c95532
18c7083c80440f2e542bf56cc5cc18156cce05efd6c2e5fea2b886c5c1e262c10af18b13ccf38c3533ba025b97bbe
62f271
545b2ab5c1f50c1dca45ce504dfcfc257362e9b43
Signature algorithm:    SHA256WithRSASignature
Key size:               2048
Issuer:                 C=US, O=IBM, OU=SKLMNode, SKLMCell, Root Certificate,
CN=c40bbc1xn3.gpfs.net
Subject:                C=US, O=IBM, OU=SKLMNode, SKLMCell, Root Certificate,
CN=c40bbc1xn3.gpfs.net

Do you trust the certificate(s) above? (yes/no) yes
To display all RKM servers information, enter:
c34f2n03:~ # mmkeyserv server show
keyserver01
        Type:                         SKLM
        Hostname:                     keyserver01.gpfs.net
        User ID:                      SKLMAdmin
        REST port:                    9080
        Label:                        1_keyserver01
        NIST:                         on
        FIPS1402:                     off
        Backup Key Servers:
        Distribute:                   yes
        Retrieval Timeout:            120
        Retrieval Retry:              3
        Retrieval Interval:           10000
        REST Certificate Expiration:  2030-12-22 21:48:53 (-0500)
        KMIP Certificate Expiration:  2021-11-02 13:03:52 (-0400)
```

   The following command displays information about all RKM servers that are known to the cluster. At the moment, the only one is `keyserver01`:

```
# mmkeyserv server show
keyserver01
        Type:                         SKLM
```

```
        Hostname:                       keyserver01.gpfs.net
        User ID:                        SKLMAdmin
        REST port:                      9080
        Label:                          1_keyserver01
        NIST:                           on
        FIPS1402:                       off
        Backup Key Servers:
        Distribute:                     yes
        Retrieval Timeout:              120
        Retrieval Retry:                3
        Retrieval Interval:             10000
        REST Certificate Expiration:  2030-12-22 21:48:53 (-0500)
        KMIP Certificate Expiration:  2021-11-02 13:03:52 (-0400)
```

2. The following command creates a tenant in the server that you defined in Example 1, `keyserver01`. The name of the tenant is `devG1`:

```
# mmkeyserv tenant add devG1 --server keyserver01
Enter password for the RKM server keyserver01:
```

   The following command displays all the current tenants of `keyserver01`:

```
# mmkeyserv tenant show
devG1
        Key Server:          keyserver01.gpfs.net
        Registered Client:   (none)
```

3. The following command adds a key client to the tenant that you created in Example 2. The command does not specify password files for the server and the new keystore, so the command prompts for the passwords. The name of the key client is `c34f2n03Client1`:

```
# mmkeyserv client create c34f2n03Client1 --server keyserver01
Enter password for the RKM server keyserver01:
Create a pass phrase for keystore:
Confirm your pass phrase:
```

   The following command displays information about all the key clients on the RKM server:

```
# mmkeyserv client show
c34f2n03Client1
        Label:                    c34f2n03Client1
        Key Server:               keyserver01.gpfs.net
        Tenants:                  (none)
        Certificate Expiration:  2021-08-20 14:20:46 (-0400)
```

4. The following command registers the key client from Example 3 to the tenant from Example 2. To ensure uniqueness in RKM IDs, it is a good practice to create the RKM ID name by combining the names of the RKM server and the tenant. However, the RKM ID cannot be longer than 21 characters. In this example the RKM ID is `keyserver01_devG1`:

```
# mmkeyserv client register c34f2n03Client1 --tenant devG1 --rkm-id keyserver01_devG1
Enter password for the RKM server :

mmkeyserv: [I] Client currently does not have access to the key.  Continue the registration
process ...
mmkeyserv: Successfully accepted client certificate
```

   The following two commands now show that key client `c34f2n03Client1` is registered to tenant `devG1`:

```
# mmkeyserv tenant show
devG1
        Key Server:          keyserver01.gpfs.net
        Registered Client:   c34f2n03Client1
        RKM ID:              keyserver01_devG1


# mmkeyserv client show
c34f2n03Client1
        Label:                    c34f2n03Client1
        Key Server:               keyserver01.gpfs.net
```

```
        Tenants:                devG1
        Certificate Expiration:  2021-08-20 14:20:46 (-0400)
```

The following command shows the contents of the new RKM stanza that was added to the RKM.conf file:

```
# mmkeyserv rkm show
keyserver01_devG1 {
  type = SKLM
  kmipServerUri = tls://192.0.2.59:5696
  keyStore = /var/mmfs/ssl/keyServ/serverKmip.1_keyserver01.c34f2n03Client1.1.p12
  passphrase = pw4c34f2n03Client1
  clientCertLabel = c34f2n03Client1
  tenantName = devG1
}
```

You can also show the contents of the RKM.conf file by routing the contents of the file to the console:

```
# cat /var/mmfs/ssl/keyServ/RKM.conf
keyserver01_devG1 {
  type = SKLM
  kmipServerUri = tls://192.0.2.59:5696
  keyStore = /var/mmfs/ssl/keyServ/serverKmip.1_keyserver01.c34f2n03Client1.1.p12
  passphrase = pw4c34f2n03Client1
  clientCertLabel = c34f2n03Client1
  tenantName = devG1
}
```

5. The following example creates an encryption key in the tenant from Example 4. In the third line, the command displays the new encryption key (KEY-d4e83148-e827-4f54-8e5b-5e1b5cc66de1):

```
# mmkeyserv key create --server keyserver01.gpfs.net --tenant devG1
Enter password for the RKM server keyserver01.gpfs.net:
KEY-d4e83148-e827-4f54-8e5b-5e1b5cc66de1
```

6. In the following example, assume that client certificate c6f2bc3n9client expires in a few days. The system administrator issues the mmkeyserv client update command to replace the old certificate with a new one that expires in 90 days.

   Note: Notice that this command also updates the keystore password. If the --keystore-pwd parameter is omitted, the keystore password remains the same.

```
# mmkeyserv client update c6f2bc3n9client --server-pwd /u/admin/README/sklm/c6f2bc3n9.pw
    --days 90 --keystore-pwd /u/admin/README/sklm/client.pw
mmkeyserv: [I] Client currently does not have access to the key. Continue the registration
    process ...
mmkeyserv: Successfully accepted client certificate
mmkeyserv: Propagating the cluster configuration data to all affected nodes. This is an
    asynchronous process.
mmkeyserv: Deleting the following KMIP certificate with label:
    2454534160085868372_vut_1578295912
Mon Jan 6 12:06:33 EST 2020: mmcommon pushSdr_async: mmsdrfs propagation started
(12:06:35) c9f1u19p1:~ # Mon Jan 6 12:06:36 EST 2020: mmcommon pushSdr_async:
    mmsdrfs propagation completed; mmdsh rc=0
```

See also

Location

/usr/lpp/mmfs/bin

mmlinkfileset command

Creates a junction that references the root directory of a GPFS fileset.

Synopsis

```
mmlinkfileset Device FilesetName [-J JunctionPath]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmlinkfileset` command creates a junction at *JunctionPath* that references the root directory of *FilesetName*. The junction is a special directory entry, much like a POSIX hard link, that connects a name in a directory of one fileset, the parent, to the root directory of a child fileset. From the user's viewpoint, a junction always appears as if it were a directory, but the user is not allowed to issue the `unlink` or `rmdir` commands on a junction. Instead, the `mmunlinkfileset` command must be used to remove a junction.

If *JunctionPath* is not specified, the junction is created in the current directory with the name *FilesetName*. The user may use the `mv` command on the directory to move to a new location in the parent fileset, but the `mv` command is not allowed to move the junction to a different fileset.

For information on GPFS filesets, see the *IBM Spectrum Scale: Administration Guide*.

Parameters

Device
The device name of the file system that contains the fileset.

File system names need not be fully-qualified. `fs0` is as acceptable as `/dev/fs0`.

FilesetName
Specifies the name of the fileset to be linked. It must not already be linked into the namespace.

There are no restrictions on linking independent filesets, but a dependent fileset can only be linked inside its own inode space.

-J *JunctionPath*
Specifies the name of the junction. The name must not refer to an existing file system object.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmlinkfileset` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

This command links fileset `fset1` in file system `gpfs1` to junction path `/gpfs1/fset1`:

```
mmlinkfileset gpfs1 fset1 -J /gpfs1/fset1
```

The system displays output similar to:

```
Fileset 'fset1' linked at '/gpfs1/fset1'.
```

To confirm the change, issue this command:

```
mmlsfileset gpfs1
```

The system displays output similar to:

```
Filesets in file system 'gpfs1':
Name                    Status    Path
root                    Linked    /gpfs1
fset1                   Linked    /gpfs1/fset1
```

See also

- “mmchfileset command” on page 224
- “mmcrfileset command” on page 311
- “mmdelfileset command” on page 369
- “mmlsfileset command” on page 501
- “mmunlinkfileset command” on page 735

Location

`/usr/lpp/mmfs/bin`

mmlsattr command

Queries file attributes.

Synopsis

```
mmlsattr [-L] [-l]
         [-d | --dump-attr]
         [-n AttributeName | --get-attr AttributeName]
         [-X | --hex-attr] [--hex-attr-name]
         [-D | --dump-data-block-disk-numbers]
         {--inode-number [SnapPath/]InodeNumber [[SnapPath/]InodeNumber...] |
          Filename[ Filename...]}
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmlsattr` command to display attributes of a file.

Results

For the specified file, the `mmlsattr` command lists:

- The current number of copies of data for a file and the maximum value
- The number of copies of the metadata for a file and the maximum value
- Whether the Direct I/O caching policy is in effect for a file
- The disk number distribution of all data block replicas for a file

Parameters

-l
Specifies that this command works only with regular files and directories and does not follow symlinks. The default is to follow symlinks.

-L
Displays additional file attributes:

- The assigned storage pool name of the file.
- The name of the fileset that includes the file.
- If a file is a snapshot file, the name of the snapshot that includes the file is shown. If the file is a regular file, an empty string is displayed.
- Whether the file is exposed, ill replicated, ill placed, or unbalanced (displayed under the `flags` heading).
- Whether the file is immutable.
- Whether the file is in `appendOnly` mode.
- The creation time of the file.
- If the `compact` attribute is nonzero, then this parameter displays the number of directory slots that were set by the `mmchattr` command with the `--compact` option or by the `gpfs_prealloc` subroutine. For more information, see the topics “mmchattr command” on page 157 and “gpfs_prealloc() subroutine” on page 958.

`-L` can be combined with `-d` | `--dump-attr` to display all extended attribute names and values for each file.

-d | --dump-attr
Displays the names of all extended attributes for each file.

-n *AttributeName* | --get-attr *AttributeName*
Displays the name and value of the specified extended attribute for each file.

-X | --hex-attr
Displays the attribute value in hex.

--hex-attr-name
Displays the attribute name in hex.

-D | --dump-data-block-disk-numbers
Displays the disk number distribution for all replicas of all data blocks for each file. It is used for resolving data block replica mismatches. For more information, see the *Replica mismatches* topic in the *IBM Spectrum Scale: Problem Determination Guide*.

--inode-number [*SnapPath*/]*InodeNumber*
The inode number of the file to be queried. You must enter at least one inode number or file name, but not both; if you specify more than one inode number, delimit each inode number by a space. If the current working directory is not already inside the active file system or snapshot, then the inode number has to be prefixed by the path to the active file system or snapshot. For example:

```
mmlsattr –inode-number /fs0/.snapshots/snap1/34608
```

You must have root authority to use this option.

Filename
The name of the file to be queried. You must enter at least one file name or inode number, but not both; if you specify more than one file name, delimit each file name by a space. Wildcard characters are supported in file names; for example, `project*.sched`.

Exit status

0
Successful completion.

nonzero
A failure has occurred. The return code equals the number of files from which the command was not able to get attribute information.

Security

You must have read access to run the `mmlsattr` command.

You may issue the `mmlsattr` command only from a node in the GPFS cluster where the file system is mounted.

Examples

1. To list the attributes of a file, issue the following command:

```
mmlsattr -L newfile
```

The system displays information as in the following example:

```
file name:            newfile
metadata replication: 1 max 2
data replication:     1 max 2
immutable:            no
appendOnly:           no
flags:                directio
storage pool name:    system
fileset name:         root
snapshot name:
creation Time:        Wed Feb 22 15:16:29 2012
```

```
Misc attributes:      ARCHIVE COMPRESSION (library lz4)
```

2. To show the attributes for all files in the root directory of file system `fs0`, issue the following command:

```
mmlsattr /fs0/*
```

The system displays information as in the following example:

```
  replication factors
metadata(max) data(max) file    [flags]
------------- --------- --------------
      1 (  1)   1 (  1) /fs0/project4.sched
      1 (  1)   1 (  1) /fs0/project4.hist
      1 (  1)   1 (  1) /fs0/project5.plan
```

3. To show all extended attribute names and values for the file `/ba1/newfile.fastq`, issue the following command:

```
mmlsattr -d -L /ba1/newfile.fastq
```

The system displays information as in the following example:

```
file name:            /ba1/newfile
metadata replication: 1 max 2
data replication:     1 max 2
immutable:            no
appendOnly:           no
flags:                directio
storage pool name:    system
fileset name:         root
snapshot name:
creation time:        Fri Jun  9 15:58:07 2017


Misc attributes:      ARCHIVE COMPRESSION (library alphae)
Encrypted:            no
user.attr1:           "value1"
user.attr:            "val1"
gpfs.DIRECTIO:        "1"
user.ea1:             "value1"


gpfs.CompressLibs:    0x0x4141616C7068616500
```

4. To show the disk number distribution of all data block replicas for the file /fs0/file1, which has 2 DataReplicas and 3 MaxDataReplicas, issue the following command:

```
mmlsattr -D /ba1/newfile.fastq
```

The system displays information as in the following example:

```


Block Index Replica 0 Replica 1 Replica 2
----------- --------- --------- ---------
0           *2         1         -
1           *3         1         -
2           *1         3         -
3           *2         3         -
4           *3         2         -
```

See also

- “mmchattr command” on page 157

Location

`/usr/lpp/mmfs/bin`

mmlscallback command

Lists callbacks that are currently registered in the GPFS system.

Synopsis

```
mmlscallback [-Y] [CallbackIdentifier[,CallbackIdentifier...] | user | system | all]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmlscallback` command to list some or all of the callbacks that are currently registered in the GPFS system.

Parameters

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

CallbackIdentifier
Indicates the callback for which information is displayed.

user
Indicates all user-defined callbacks. This is the default.

system
Indicates all system-defined callbacks.

all
Indicates all callbacks currently registered with the system.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmlscallback` command

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

To list all of the callbacks that are currently in the GPFS system, issue this command:

```
mmlscallback
```

The system displays information similar to:

```
test1
        command       = /tmp/myScript
        event         = startup

test2
        command       = /tmp/myScript2
        event         = shutdown
        parms         = %upNodes
```

To list a specific callback (for example, **test2**) that is currently in the GPFS system, issue this command:

```
mmlscallback test2
```

The system displays information similar to:

```
test2
        command       = /tmp/myScript2
        event         = shutdown
        parms         = %upNodes
```

See also

- “mmaddcallback command” on page 12
- “mmdelcallback command” on page 362

Location

/usr/lpp/mmfs/bin

mmlscluster command

Displays the current configuration information for a GPFS cluster.

Synopsis

```
mmlscluster [-Y] [--ces] [--cnfs] [--cloud-gateway]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmlscluster` command to display the current configuration information for an IBM Spectrum Scale cluster.

For the IBM Spectrum Scale cluster, the `mmlscluster` command displays:

- The cluster name
- The cluster ID
- The UID domain
- The remote shell command being used
- The remote file copy command being used
- The repository type (CCR or server-based)
- The primary cluster configuration server (if server-based repository)
- The secondary cluster configuration server (if server-based repository)
- A list of nodes belonging to the IBM Spectrum Scale cluster

For each node, the command displays:

- The node number assigned to the node by IBM Spectrum Scale
- GPFS daemon node interface name
- Primary network IP address
- IBM Spectrum Scale administration node interface name
- Designation, such as whether the node is any of the following:
  - quorum node - A node in the cluster that is counted to determine if a quorum exists. Members of a cluster use the quorum node to determine if it is safe to continue I/O operations when a communications failure occurs.
  - manager node - The file system node that provides the file system manager services to all of the nodes using the file system, including: file system configuration, disk space allocation, token management, and quota management.
  - snmp_collector node - The designated SNMP collector node for the cluster. The GPFS SNMP subagent runs on the designated SNMP collector node. For additional information, see *GPFS SNMP support* in *IBM Spectrum Scale: Problem Determination Guide*.
  - gateway node - Ensures primary and Disaster Recovery cluster communication during failover.
  - perfmon node - Performance monitoring nodes collect metrics and performance information and sends the information to one or more performance collection nodes.

Parameters

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

--ces
Displays information about protocol nodes.

--cnfs
Displays information about clustered NFS.

--cloud-gateway
Displays information about Transparent cloud tiering nodes.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmlscluster` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To display the current configuration information for the GPFS cluster, issue the **mmlscluster** command with no parameters:

```
# mmlscluster
```

   A sample output is as follows:

```
GPFS cluster information
========================
  GPFS cluster name:         samplecluster.sample.com
  GPFS cluster id:           13953039421150668925
  GPFS UID domain:           samplecluster.sample.com
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR

Node  Daemon node name     IP address     Admin node name       Designation
-------------------------------------------------------------------------------
   1  node1.sample.com     192.168.1.1    node1.sample.com      quorum-manager
   2  node2.sample.com     192.168.1.2    node2.sample.com      quorum-manager
   3  node3.sample.com     192.168.1.3    node3.sample.com      quorum-manager
   4  node4.sample.com     192.168.1.4    node4.sample.com
   5  node5.sample.com     192.168.1.5    node4.sample.com
   6  node6.sample.com     192.168.1.6    node4.sample.com
```

2. To display the configuration information about the Transparent cloud tiering nodes, issue the following command:

```
# mmlscluster --cloud-gateway
```

A sample output is as follows:

```
GPFS cluster information
========================
  GPFS cluster name:         c350f1u1b11
  GPFS cluster id:           9364209917238477017

 Node  Daemon node name               Cloud node type
-------------------------------------------------
   1   c350f1u1b11                     Cloud-Gateway
   2   c350f8u17.pk.labs.ibm.com       Cloud-Gateway
   3   c350f8u18.pk.labs.ibm.com       Cloud-Gateway
```

See also

- “mmaddnode command” on page 34
- “mmchcluster command” on page 165
- “mmcrcluster command” on page 306
- “mmdelnode command” on page 375

Location

/usr/lpp/mmfs/bin

mmlsconfig command

Displays the current configuration data for a GPFS cluster.

Synopsis

```
mmlsconfig [Attribute[,Attribute...]] [-Y]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmlsconfig` command to display the requested configuration attributes for a GPFS cluster. If no specific attributes are requested, the command displays all values that were set explicitly by the user. Depending on your configuration, additional information that is set by GPFS might be displayed. If a configuration attribute is not shown in the output of this command, the default value for that attribute, as documented in the `mmchconfig` command, is in effect.

Parameters

Attribute
Specifies the name of an attribute to display with its value. If no name is specified, the command displays a default list of attributes with their values. See Example 1.

For descriptions of the attributes, see the topic "mmchconfig command" on page 170. The `mmlsconfig` command, which lists the values of attributes, and the `mmchconfig` command, which sets the values of attributes, use the same attribute names.

Exception: To update the minimum release level, issue the `mmchconfig` command with the attribute **`release`**=**`LATEST`**. To display the value of the minimum release level, issue the `mmlsconfig` command with the **`minReleaseLevel`** parameter. For more information, see the topic *Minimum release level of a cluster* in the *IBM Spectrum Scale: Administration Guide*.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **`mmclidecode`**. Use the **`mmclidecode`** command to decode the field.

Exit status

0
Successful completion.

nonzero
A failure occurred.

Security

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster. It must be able to do so without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To display the current configuration data for the GPFS cluster that you are running on, issue this command:

```
mmlsconfig
```

   The system displays information similar to the following example:

```
Configuration data for cluster small.cluster:
---------------------------------------------
myNodeConfigNumber 1
clusterName small.cluster
clusterId 6339012640885012929
autoload yes
minReleaseLevel 4.2.0.0
dmapiFileHandleSize 32
[c6f1c3vp3]
pagepool 512M
[common]
adminMode central

File systems in cluster small.cluster:
--------------------------------------
/dev/fs1
/dev/gpfs1
```

2. To display the current values for the `maxblocksize` and `pagepool` attributes, issue the following command:

```
mmlsconfig maxblocksize,pagepool
```

   The system displays information similar to the following example:

```
maxblocksize 4M
pagepool 1G
pagepool 512M [c6f1c3vp3]
```

3. To display the current value for the `cipherList` attribute, issue this command:

```
mmlsconfig cipherList
```

   The system displays information similar to the following example:

```
cipherList AUTHONLY
```

See also

- “mmchcluster command” on page 165
- “mmchconfig command” on page 170
- “mmcrcluster command” on page 306

Location

`/usr/lpp/mmfs/bin`

mmlsdisk command

Displays the current configuration and state of the disks in a file system.

Synopsis

```
mmlsdisk Device [-d "DiskName[;DiskName...]"] [-e | -Y] [-L]
```

or

```
mmlsdisk Device [-d "DiskName[;DiskName...]"] {-m | -M} [-Y]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmlsdisk` command to display the current state of the disks in the file system.

The `mmlsdisk` command may be run against a mounted or unmounted file system.

For each disk in the list, the `mmlsdisk` command displays the following:

- Disk name
- Driver type
- Logical sector size (under the heading "sector size")
- Failure group
- Whether it holds metadata
- Whether it holds data
- Status:

  ready
  Normal status.

  suspended
  or
  to be emptied
  Indicates that data is to be migrated off this disk.

  being emptied
  Transitional status in effect while a disk deletion is pending.

  emptied
  Indicates that data is already migrated off this disk.

  replacing
  Transitional status in effect for old disk while replacement is pending.

  replacement
  Transitional status in effect for new disk while replacement is pending.
- Availability:

  up
  The disk is available to GPFS for normal **read** and **write** operations.

  down
  No **read** and **write** operations can be performed on this disk.

recovering
An intermediate state for disks coming up, during which GPFS verifies and corrects data. **write** operations can be performed while a disk is in this state, but **read** operations cannot (because data on the disk being recovered might be stale until the `mmchdisk start` command completes).

unrecovered
The disk was not successfully brought up.

- Disk ID
- Storage pool to which the disk is assigned
- Remarks: A tag is displayed if the disk is a file system descriptor replica holder, an excluded disk, or the disk supports space reclaim.

Parameters

Device
The device name of the file system to which the disks belong. File system names need not be fully-qualified. `fs0` is as acceptable as `/dev/fs0`.

This must be the first parameter.

-d "*DiskName*[;*DiskName*...]"
The name of the disks for which you want to display current configuration and state information. When you enter multiple values for *DiskName*, separate them with semicolons and enclose the list in quotation marks.

```
"gpfs3nsd;gpfs4nsd;gpfs5nsd"
```

Options

-e
Displays all of the disks in the file system that do not have an availability of **up** and a status of **ready**. If all disks in the file system are **up** and **ready**, the message displayed is:

```
6027-623 All disks up and ready
```

-L
Displays an extended list of disk parameters that includes the `disk id` column and the `remarks` column. The `remarks` column can contain one or more of the following tags:

desc
The disk is a file system descriptor replica holder.

excl
The disk is excluded by the `mmfsctl` command.

{nvme(t) | scsi(t) | auto(t)}
The disk is a device support space reclaim:

nvme(t)
The disk is a TRIM capable NVMe device that supports the `mmreclaimspace` command.

scsi(t)
The disk is a thin provisioned SCSI disk that supports the `mmreclaimspace` command.

auto(t)
The disk is either an `nvme(t)` device or a `scsi(t)` device. IBM Spectrum Scale will try to detect the actual disk type automatically. To avoid problems, you should replace `auto` with the correct disk type, `nvme` or `scsi`, as soon as you can.

Note: In 5.0.5, the space reclaim auto-detection is enhanced. It is encouraged to use the auto key-word after your cluster is upgraded to 5.0.5.

For more information, see the topic *IBM Spectrum Scale with data reduction storage devices* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

-M
Displays whether I/O requests to the disk are satisfied on the local node, or using an NSD server. If the I/O is done using an NSD server, shows the NSD server name and the underlying disk name on that server node.

-m
Displays whether I/O requests to the disk are satisfied on the local node, or using an NSD server. The scope of this option is the node on which the `mmlsdisk` command is issued.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

If you are a root user, the node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

As root, the command can also do an `mmlsdisk` on remote file systems.

If you are a non-root user, you may specify only file systems that belong to the same cluster as the node on which the `mmlsdisk` command was issued.

The `mmlsdisk` command does not work if GPFS is down.

Examples

1. To display the current state of **gpfs2nsd**, issue the following command:

```
# mmlsdisk /dev/fs0 -d gpfs2nsd
```

A sample output is as follows:

```
disk      driver  sector failure holds    holds                     storage
name      type      size   group metadata data  status  availability pool
--------  ------- ------ ------- -------- ----- ------- ------------ -------
gpfs2nsd  nsd        512    4002 yes      yes   ready   up           system
```

Note: In this output, "sector size" refers to logical sector size.

2. To display the current states of `gpfs2nsd`, `gpfs3nsd`, and `gpfs4nsd`, and display their respective disk ids and the descriptor quorum assignment, issue the following command:

```
# mmlsdisk /dev/fs0 -d "gpfs2nsd;gpfs3nsd;gpfs4nsd" -L
```

A sample output is as follows:

```
disk      driver sector failure holds    holds                            storage
name      type     size   group metadata data  status availability disk id pool     remarks
--------- ------ ------ ------- -------- ----- ------ ------------ ------- -------- -------
gpfs2nsd  nsd      512    4002  yes      yes   ready  up           2       system    desc
gpfs3nsd  nsd      512    4002  yes      yes   ready  up           3       system
gpfs4nsd  nsd      512    4002  yes      yes   ready  up           4       system
```

```
Number of quorum disks: 3
Read quorum value:      2
Write quorum value:     2
```

Note: In this output, "sector size" refers to logical sector size.

3. After the `mmchdisk fs0 empty -d gpfs1nsd` command has been issued, you can view the current state of `gpfs1nsd` by issuing the following command:

```
# mmlsdisk fs0 -L
```

A sample output from IBM Spectrum Scale 4.1.1 and later is as follows:

```
disk     driver sector failure holds   holds                                    storage
name     type   size   group  metadata data  status     availability  disk id pool
-------- ------ ------ ------ -----    -----  -------    ------------- ------- -----
gpfs1nsd nsd     512      -1  Yes      Yes   to be emptied up              1  system
gpfs2nsd nsd     512      -1  Yes      Yes   to be emptied up              2  system
gpfs3nsd nsd     512      -1  Yes      Yes   ready         up              3  system
gpfs4nsd nsd     512      -1  Yes      Yes   ready         up              4  system
Number of quorum disks: 3
Read quorum value:      2
Write quorum value:     2
Attention: Due to an earlier configuration change the file system
may contain data that is at risk of being lost.
```

4. To display whether the I/O is performed locally or using an NSD server, the NSD server name, and the underlying disk name for the file system named `test`, issue the following command:

```
# mmlsdisk test -M
```

A sample output is as follows:

```
Disk name     IO performed on node     Device             Availability
------------  -----------------------  -----------------  ------------
gpfs7nsd      localhost                /dev/hdisk12       up
gpfs10nsd     k5n88.kgn.ibm.com        /dev/hdisk13       up
gpfs4nsd      localhost                /dev/hdisk10       up
```

5. To display the same information as in the previous example, but limited to the node on which the command is issued, issue the following command:

```
# mmlsdisk test -m
```

A sample output is as follows:

```
Disk name     IO performed on node     Device             Availability
------------  -----------------------  -----------------  ------------
gpfs7nsd      localhost                /dev/hdisk12       up
gpfs10nsd     k5n88.kgn.ibm.com              -            up
gpfs4nsd      localhost                /dev/hdisk10       up
```

See also

- “mmadddisk command” on page 28
- “mmchdisk command” on page 212
- “mmdeldisk command” on page 364
- “mmrpldisk command” on page 690

Location

`/usr/lpp/mmfs/bin`

mmlsfileset command

Displays attributes and status for GPFS filesets.

Synopsis

```
mmlsfileset Device
      [[Fileset[,Fileset...]] [-J Junction[,Junction...]] | -F FileName]
      [-d [--block-size {BlockSize | auto}]] [-i] [-L] [-X] [-Y]
      [--afm] [--deleted] [--iam-mode]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmlsfileset` command to display information for the filesets that belong to a given GPFS file system. The default is to display information for all filesets in the file system. You may choose to display information for only a subset of the filesets.

The operation of the `-L` flag omits the attributes listed without it, namely status and junction path. In addition, if the fileset has status `Deleted`, then `-L` also displays the name of the latest snapshot that includes the fileset in place of the root inode number and parent fileset identifier.

The attributes displayed are:

- Name of the fileset
- Status of the fileset (when the `-L` flag is omitted)
- Junction path to the fileset (when the `-L` flag is omitted)
- Fileset identifier (when the `-L` flag is included)
- Root inode number, if not deleted (when the `-L` flag is included)
- Parent fileset identifier, if not deleted (when the `-L` flag is included)
- Latest including snapshot, if deleted (when the `-L` flag is included)
- Creation time (when the `-L` flag is included)
- Inode space (when the `-L` flag is included)
- Number of inodes in use (when the `-i` flag is included)
- Data size (when the `-d` flag is included)
- Comment (when the `-L` flag is included)
- Caching-related information (when the `--afm` flag is included)
- Value of the permission change flag (when the `-X` flag is used to generate stanza output)
- Integrated archive manger (IAM) mode information

For information on GPFS filesets, see *Filesets* in the *IBM Spectrum Scale: Administration Guide*.

Parameters

Device
The device name of the file system that contains the fileset.

File system names need not be fully-qualified. `fs0` is as acceptable as `/dev/fs0`.

This must be the first parameter.

Fileset
Specifies a comma-separated list of fileset names.

-J *Junction*
Specifies a comma-separated list of path names. They are not restricted to fileset junctions, but may name any file or directory within the filesets to be listed.

Note: The base of the junction path that is displayed is always the default mount point of the file system in the cluster that owns the file system. For example, suppose that a file system has a default mount point of `/fs0` in the owning cluster and a default mount point of `/remote_fs0` in a remote cluster that accesses the file system. If you issue the `mmlsfileset` command on the accessing cluster `/remote_fs0`, the junction path that is displayed begins with `/fs0`, not `/remote_fs0`.

-F *FileName*
Specifies the name of a file containing either fileset names or path names. Each line must contain a single entry. All path names must be fully-qualified.

-d
Displays the amount of storage in use for the fileset.

This operation requires an amount of time that is proportional to the size of the file system; therefore, it can take several minutes or even hours on a large and heavily-loaded file system.

This optional parameter can impact overall system performance. Avoid running the `mmlsfileset` command with this parameter frequently or during periods of high file system activity.

This option is not valid if the *Device* parameter is a remote file system.

--block-size {*BlockSize* | auto}
Specifies the unit in which the number of blocks is displayed. The value must be of the form [*n*]K, [*n*]M, [*n*]G or [*n*]T, where *n* is an optional integer in the range 1 to 1023. The default is 1K. If `auto` is specified, the number of blocks is automatically scaled to an easy-to-read value.

-i
Displays the number of inodes in use for the fileset.

This option is not valid if the *Device* parameter is a remote file system.

This operation requires an amount of time that is proportional to the number of inodes in the file system; therefore, it can take several minutes or even hours on a large and heavily-loaded file system.

Information about the number of inodes in the fileset can be retrieved more efficiently with the following command, if quota management has been enabled for the file system:

```
mmrepquota -j FileSystem
```

-L
Displays additional information for the fileset. This includes:

- Fileset identifier
- Root inode number
- Parent identifier
- Fileset creation time
- Inode space
- User defined comments, if any

If the fileset is a dependent fileset, dpnd will be displayed next to the inode space identifier.

-X
Generates stanza output containing the following:

- The same information presented by the `-L` flag
- The value of the permission change flag
- The same information presented by the `--afm` flag

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

--afm
Displays caching-related information for the fileset.

--deleted
Displays only the filesets with a status of `Deleted`.

--iam-mode
Displays integrated archive manager (IAM) mode information. For more information, see "mmchfileset command" on page 224.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

Fileset owners can run the `mmlsfileset` command with the `-L`, `-d`, and `-i` options.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. This command displays fileset information for all filesets in file system `gpfs1`:

```
# mmlsfileset gpfs1
```

A sample output is as follows:

```
Filesets in file system 'gpfs1':
Name                    Status    Path
root                    Linked    /gpfs1
fset1                   Linked    /gpfs1/fset1
fset2                   Linked    /gpfs1/fset1/fset2
```

2. These commands display information for a file system with filesets and snapshots. Note that deleted filesets that are saved in snapshots are displayed with the name enclosed in parentheses.

   a. Command:

```
# mmlsfileset fs1 -d -i
```

A sample output is as follows:

```
Filesets in file system 'fs1':
Name    Status   Path                         Inodes  Data (in  KB)
root    Linked   /gpfs                           3           53528
(gone)  Deleted  /gpfs/.snapshots/Snap17/gone   0               0
TestF4  Linked   /gpfs/test-f4                   3              24
TestF3  Linked   /gpfs/test-f4/dir1/f3           2              16
TestF2  Unlinked  --                            98             784
TestF5  Linked   <TestF2>/subdir/f5              1               8
```

b. Command:

```
# mmlsfileset fs1 --deleted
```

A sample output is as follows:

```
Filesets in file system 'fs1':
Name    Status   Path
(gone)  Deleted  /gpfs/.snapshots/Snap17/gone
```

c. Command:

```
# mmlsfileset fs1 --afm
```

A sample output is as follows:

```
Filesets in file system 'fs1':
Name                     Status    Path
afmTarget
root                     Linked    /gpfs/fs1
--
ro1                      Linked    /gpfs/fs1/ro1                          hs21n45:/
gpfs/fs1/ro1
sw1                      Linked    /gpfs/fs1/sw1                          hs21n45:/
gpfs/fs1/sw1
lu1                      Linked    /gpfs/fs1/lu1                          hs21n45:/
gpfs/fs1/lu1
```

d. Command:

```
# mmlsfileset fs1 -L
```

A sample output is as follows:

```
Filesets in file system 'fs1':
Name           Id    RootInode  ParentId Created                  InodeSpace  MaxInodes  AllocInodes
Comment
root            0            3        -- Mon Jan 23 18:59:36 2012    0          1000064        65792
root fileset
TestF4          2        59446         0 Wed Feb  1 09:28:50 2012    0                0            0
4th in series
TestF3          3        59435         2 Wed Feb  1 09:28:52 2012    0                0            0
(gone)          1 latest:         Snap17 Wed Feb  1 09:28:46 2012    0                0            0
Not forgotten
TestF2          4        59437        -- Wed Feb  1 09:28:52 2012    0                0            0
TestF5          5         7017         4 Wed Feb  1 09:28:53 2012    0                0            0
Number 5
FsetF1-V2       6       131075        -- Wed Feb  1 09:28:55 2012    1           100096       100096
FsetF2-V2       7       262147        -- Wed Feb  1 09:28:56 2012    2           100096       100096
FsetF2-V2-lite  9       263680        -- Wed Feb  1 09:28:59 2012    2 dpnd           0            0
FsetF3-V2       8       393219        -- Wed Feb  1 09:28:57 2012    3           100096       100096
```

e. Command:

```
# mmlsfileset fs1 sw1 --afm -L
```

A sample output is as follows:

```
Filesets in file system 'fs1':


Attributes for fileset sw:
==========================
Status                              Linked
Path                                /gpfs/fs1/sw
afm-associated                      Yes
Target                              c2m3n06:/gpfs/fs2
Mode                                single-writer
File Lookup Refresh Interval        30 (default)
File Open Refresh Interval          30 (default)
Dir Lookup Refresh Interval         60 (default)
Dir Open Refresh Interval           60 (default)
Async Delay                         15 (default)
Expiration Timeout                  disable
Recovery Point Objective            disable (default)
```

```
Last pSnapId                        0
Display Home Snapshots              no
```

f. Command:

```
# mmlsfileset fs1 TestF2,TestF5 -J /gpfs/test-f4/dir1,/gpfs/test-f4/dir1/f3/dir2/
```

A sample output is as follows:

```
Filesets in file system 'fs1':
Name                    Status    Path
TestF2                  Unlinked  --
TestF5                  Linked    <TestF2>/subdir/f5
TestF4                  Linked    /gpfs/test-f4
TestF3                  Linked    /gpfs/test-f4/dir1/f3
```

g. Command:

```
# mmlsfileset gpfsha --deleted
```

A sample output is as follows:

```
Filesets in file system 'gpfsha':
Name                    Status    Path
(fset17)                Deleted   /gpfsha/.snapshots/snap20/fset17
```

See also

- “mmchfileset command” on page 224
- “mmcrfileset command” on page 311
- “mmdelfileset command” on page 369
- “mmlinkfileset command” on page 485
- “mmunlinkfileset command” on page 735

Location

/usr/lpp/mmfs/bin

mmlsfs command

Displays file system attributes.

Synopsis

```
mmlsfs {Device | all | all_local | all_remote} [-A] [-B] [-d] [-D]
       [-E] [-f] [-i] [-I] [-j] [-k] [-K] [-L] [-m] [-M] [-n] [-o]
       [-P] [-Q] [-r] [-R] [-S] [-t] [-T] [-V] [-Y] [-z]
       [--create-time] [--encryption] [--fastea] [--file-audit-log]
       [--filesetdf] [--inode-limit] [--is4KAligned] [--log-replicas]
       [--maintenance-mode] [--mount-priority] [--perfileset-quota]
       [--rapid-repair] [--subblocks-per-full-block] [--write-cache-threshold]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmlsfs` command to list the attributes of a file system. You can issue the `mmlsfs` command for file systems that belong to the current cluster or for file systems that are owned by other clusters.

Depending on your configuration, additional information that is set by GPFS can be displayed to help problem determination when contacting the IBM Support Center.

Results

If you do not specify any options, all attributes of the file system are displayed. When you specify options, only those attributes that are specified are listed, in the order issued in the command. Some parameters are preset for optimum performance and, although they are displayed in the `mmlsfs` command output, you cannot change them.

Parameters

The following parameter must be the first parameter:

***Device* | all | all_local | all_remote**

Device
: Indicates the device name of the file system for which information is displayed. File system names do not need to be fully qualified. `fs0` is as acceptable as `/dev/fs0`.

all
: Indicates all file systems that are known to this cluster.

all_local
: Indicates all file systems that are owned by this cluster.

all_remote
: Indicates all file systems that are owned by another cluster.

This must be the first parameter.

The following optional parameters, when used, must be provided after the *Device* | `all` | `all_local` | `all_remote` parameter:

-A
: Displays if and when the file system is automatically mounted.

-B
: Displays the block size of the file system in bytes. For more information about block size, see the description of the `-B` *BlockSize* parameter in “mmcrfs command” on page 318.

-d
Displays the names of all of the disks in the file system.

-D
Displays the type of file locking semantics that are in effect (`nfs4` or `posix`).

-E
Displays the exact `mtime` values reported.

-f
Displays the minimum fragment (subblock) size of the file system in bytes. The subblock size and the number of subblocks in a block are determined by the block size. For more information, see the description of the `-B` *BlockSize* parameter in "mmcrfs command" on page 318.

-i
Displays the inode size, in bytes.

-I
Displays the indirect block size, in bytes.

-j
Displays the block allocation type.

-k
Displays the type of authorization that is supported by the file system.

-K
Displays the strict replication enforcement.

-L
Displays the internal log file size.

-m
Displays the default number of metadata replicas.

-M
Displays the maximum number of metadata replicas.

-n
Displays the estimated number of nodes for mounting the file system.

-o
Displays the additional mount options.

-P
Displays the storage pools that are defined within the file system.

-Q
Displays which quotas are currently enforced on the file system.

-r
Displays the default number of data replicas.

-R
Displays the maximum number of data replicas.

-S
Displays whether the updating of `atime` is suppressed for the `gpfs_stat()`, `gpfs_fstat()`, `stat()`, and `fstat()` calls.

-t
Displays the Windows drive letter.

-T
Displays the default mount point.

-V
Displays the current format version of the file system.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

-z
Displays whether DMAPI is enabled for this file system.

--create-time
Displays the creation time of the file system.

--encryption
Displays a yes or no value to indicate whether encryption is enabled. This value cannot be changed with the mmchfs command. When the cluster is created this value is set to no. When an encryption policy is established for the file system, the value is set to yes.

--fastea
Displays a yes or no value to indicate whether fast external attributes are enabled. Displays a migrating value if migration was initiated with mmmigratefs --fastea but is not yet complete.

--file-audit-log
Displays whether file audit logging is enabled or disabled.

--filesetdf
Displays a yes or no value to indicate whether filesetdf is enabled. If yes, the df command reports either quota limit and usage or inode space capacity and usage for the fileset and not for the total file system. This option affects the df command behavior only on Linux nodes.

The df command reports quota limit and quota usage if quota is enabled for the fileset. If quota is disabled and filesetdf is enabled in IBM Spectrum Scale 5.1.1 or later with file system version 5.1.1 or later, then the **df** command reports inode space capacity and inode usage at the independent fileset-level. However, the **df** command reports the block space at the file system-level because the block space is shared with the whole file system.

Note: If quota is enabled, then no behavior change for **df** command with **filesetdf** enabled, regardless of the cluster and file system versions.

--inode-limit
Displays the maximum number of files in the file system.

--is4KAligned
Displays whether file systems are formatted to be 4K aligned.

--log-replicas
Displays the number of recovery log replicas. If a value of 0 is displayed, the number of recovery log replicas is the same as the number of metadata replicas currently in effect for the file system.

--maintenance-mode
Displays a yes or no value to indicate whether file system maintenance mode is on or off:

- The value yes indicates that file system maintenance mode is on.
- The value no, which is the default, indicates that file system maintenance mode is off.

For more information on file system maintenance mode, see *File system maintenance mode* in *IBM Spectrum Scale: Administration Guide*.

Note: Another possible output value can be request in progress. This output value means that the file system has been asked to enable or disable file system maintenance mode, and that request is in progress.

--mount-priority
Displays the assigned mount priority.

--perfileset-quota
Displays the per-fileset quota.

--rapid-repair
Displays a yes or no value indicating whether the per-block replication tracking and repair feature is enabled.

--write-cache-threshold
Displays the threshold below which synchronous writes are initially buffered in the highly available write cache before being written back to primary storage.

--subblocks-per-full-block
Displays the number of subblocks in a file system data block.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Note: The command treats the following conditions as failures:

- The file system that you specified was not found.
- You specified `all`, `all_local`, or `all_remote` and no file systems were found.

Security

If you are a root user, the node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

The following command displays the attributes of the file system `gpfs1`:

```
mmlsfs gpfs1
```

The system displays information like the following:

```
flag                value                    description
------------------- ------------------------ -----------------------------------
 flag                     value                          description
-------------------           ------------------------
-----------------------------------
-f                         8192                           Minimum fragment (subblock) size in
bytes
-i                         4096                           Inode size in bytes
-I                         32768                          Indirect block size in bytes
-m                         1                              Default number of metadata replicas
-M                         2                              Maximum number of metadata replicas
-r                         1                              Default number of data replicas
-R                         2                              Maximum number of data replicas
-j                         cluster                        Block allocation type
-D                         nfs4                           File locking semantics in effect
-k                         all                            ACL semantics in effect
-n                         32                             Est. num. of nodes that will mount
file system
-B                         4194304                        Block size
-Q                         none                           Quotas accounting enabled
                           none                           Quotas enforced
                           none                           Default quotas enabled
--perfileset-quota         No                             Per-fileset quota enforcement
--filesetdf                No                             Fileset df enabled?
-V                         19.01 (5.0.1.0)                File system version
-z                         No                             Is DMAPI enabled?
-L                         33554432                       Logfile size
-E                         Yes                            Exact mtime mount option
-S                         relatime                       Suppress atime mount option
-K                         whenpossible                   Strict replica allocation option
--fastea                   Yes                            Fast external attributes enabled?
--encryption               No                             Encryption enabled?
--inode-limit              590848                         Maximum number of inodes
--log-replicas             0                              Number of log replicas
--is4KAligned              Yes                            is4KAligned?
--rapid-repair             Yes                            rapidRepair enabled?
```

```
--write-cache-threshold    0                               HAWC Threshold (max 65536)
--subblocks-per-full-block 512                             Number of subblocks per full block
-P                         system                          Disk storage pools in file system
--file-audit-log           No                              File Audit Logging enabled?
--maintenance-mode         No                              Maintenance Mode enabled?
-d                         mpathf;mpathh                   Disks in file system
-A                         yes                             Automatic mount option
-o                         none                            Additional mount options
-T                         /gpfs/gpfs1                     Default mount point
--mount-priority           0                               Mount priority
```

If you issue the `mmlsfs` command with the **all** option:

```
mmlsfs all -A
```

The system displays information similar to:

```
File system attributes for /dev/fs1:
====================================
flag                value                    description
------------------- ------------------------ -----------------------------------
 -A                 yes                      Automatic mount option

File system attributes for /dev/gpfs1:
======================================
flag                value                    description
------------------- ------------------------ -----------------------------------
 -A                 yes                      Automatic mount option
```

See also

- “mmcrfs command” on page 318
- “mmchfs command” on page 232
- “mmdelfs command” on page 373

Location

`/usr/lpp/mmfs/bin`

mmlslicense command

Displays information about the IBM Spectrum Scale node licensing designation or about disk and cluster capacity.

Synopsis

```
mmlslicense [-Y] [-L | --capacity [--formatted] | --licensed-usage|--ilmt-data]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmlslicense` command to display the number of IBM Spectrum Scale client, FPO, and server licenses assigned to the nodes in the cluster.

For information on IBM Spectrum Scale license designation, see *IBM Spectrum Scale license designation* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Parameters

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

-L
Displays information about the license type that is associated with each of the nodes in the cluster. An asterisk after the license type indicates insufficient license level for the roles that the node performs.

--capacity [--formatted]
Displays disk and cluster size information.

--formatted
Inserts commas to separate groups of three numerals for readability.

--licensed-usage
Displays the names and sizes of the NSDs, the total size of all the NSDs, and the licensed capacity limit. See Example 4. This option is valid only in the IBM Spectrum Scale Developer Edition.

Note: The licensed capacity of the IBM Spectrum Scale Developer Edition is 12 TB (12,000,000,000,000 bytes) of storage. The `mmcrnsd` command calculates the size of the current NSDs plus the size of the proposed new NSDs and fails with an error message if the licensed capacity would be exceeded. For more information, see the "mmcrnsd command" on page 335.

--ilmt-data
Writes the software identify information such as the product edition and the product ID into the `/var/adm/ras/ILMT_Data.slmtag`. This information is used by the IBM License Metric Tool (ILMT). The option also writes the DECIMAL_TERABYTE metric into the log file. This metric represents the terabytes of storage capacity, in decimal number, for all the NSDs in the Cluster.

This option must be used independently and not with any other option, including the `-Y` option. It is valid only in the following editions:

- IBM Spectrum Scale Erasure Code Edition
- IBM Spectrum Scale Data Management Edition

- IBM Spectrum Scale Data Access Edition

Exit status

0
Successful completion.

nonzero
A failure occurred.

Security

You must have root authority to run the `mmlslicense` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster. The shell commands must be executed without the use of a password and must not produce any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. The following command displays summary information about the IBM Spectrum Scale licenses of the nodes in the cluster:

```
#mmlslicense
Summary information
---------------------
Number of nodes defined in the cluster:                         4
Number of nodes with server license designation:                 1
Number of nodes with FPO license designation:                   0
Number of nodes with client license designation:                 2
Number of nodes still requiring server license designation:      1
Number of nodes still requiring client license designation:      1
This node runs IBM Spectrum Scale Advanced Edition
```

2. The following command displays the types of IBM Spectrum Scale licenses that are associated with the nodes in the cluster:

```
#mmlslicense -L
Node name                    Required license   Designated license
--------------------------------------------------------------------
k145n05.kgn.ibm.com              server            server
k145n06.kgn.ibm.com              server            client  *
k145n07.kgn.ibm.com              client            client
k145n08.kgn.ibm.com              client            none    *

 Summary information
---------------------
Number of nodes defined in the cluster:                         4
Number of nodes with server license designation:                 1
Number of nodes with FPO license designation:                   0
Number of nodes with client license designation:                 2
Number of nodes still requiring server license designation:      1
Number of nodes still requiring client license designation:      1
This node runs IBM Spectrum Scale Advanced Edition
```

3. The following command displays information for capacity-based licensing:

```
#mmlslicense --capacity --formatted
NSD Summary:
======================
Total Number of NSDs:  2
mynsd1:                           10,737,418,240 Bytes
mynsd2:                           10,737,418,240 Bytes

Cluster Summary:
======================
Cluster Total Capacity:   21,474,836,480 Bytes
```

4. The following command displays the names and sizes of the NSDs, the total size of all the NSDs, and the licensed capacity limit:

```
#mmlslicense --licensed-usage
 NSD Name                                          NSD Size (Bytes)
-------------------------------------------------------------------
 de1_c35f1m4n09_sda                                 299,439,751,168
 de1_c35f1m4n09_sdf                                  99,488,890,880
 de1_mmimport_test                                   73,284,976,640


 Developer Edition Summary:
 ==========================================================
 Total NSD Licensed Limit (12 TB):      12,000,000,000,000 (Bytes)
 Total NSD Size:                           472,213,618,688 (Bytes)
```

5. The following command displays disk capacities in an environment that contains both vdisks and physical disks:

```
# mmlslicense --capacity

 NSD Summary:
 ======================
  Total Number of NSDs:     40
  RG001LG001VS002:          53957623808 Bytes
  RG001LG002VS002:          53957623808 Bytes
  RG001LG003VS002:          53957623808 Bytes
  RG001LG004VS002:          53957623808 Bytes
  RG001LG005VS002:          53957623808 Bytes
  RG001LG006VS002:          53957623808 Bytes
  RG002LG001VS016:          16544432128 Bytes
  RG002LG002VS016:          16544432128 Bytes
  RG002LG003VS016:          16544432128 Bytes
  RG002LG004VS016:          16544432128 Bytes
  RG002LG005VS016:          16544432128 Bytes
  RG003LG001VS001:          2002248531968 Bytes
  RG003LG001VS004:          252160507904 Bytes
  RG003LG001VS007:          252160507904 Bytes
  RG003LG002VS001:          2002248531968 Bytes
  RG003LG002VS004:          252160507904 Bytes
  RG003LG002VS007:          252160507904 Bytes
  RG003LG003VS001:          2002248531968 Bytes
  RG003LG003VS004:          252160507904 Bytes
  RG003LG003VS007:          252160507904 Bytes
  RG003LG004VS001:          2002248531968 Bytes
  RG003LG004VS004:          252160507904 Bytes
  RG003LG004VS007:          252160507904 Bytes
  RG003LG005VS001:          2002248531968 Bytes
  RG003LG005VS004:          252160507904 Bytes
  RG003LG005VS007:          252160507904 Bytes
  RG003LG006VS001:          2002248531968 Bytes
  RG003LG006VS004:          252160507904 Bytes
  RG003LG006VS007:          252160507904 Bytes
  RG003LG006VS001:          2002248531968 Bytes
  RG003LG006VS004:          252160507904 Bytes
  RG003LG006VS007:          252160507904 Bytes
  RG003LG007VS001:          2002248531968 Bytes
  RG003LG007VS004:          252160507904 Bytes
  RG003LG007VS007:          252160507904 Bytes
  RG003LG008VS001:          2002248531968 Bytes
  RG003LG008VS007:          252160507904 Bytes
  sdb:                      2097152000 Bytes
  sdc:                      2097152000 Bytes
  sdd:                      2097152000 Bytes
  sde:                      2097152000 Bytes
  sdf:                      2097152000 Bytes
  sdg:                      2097152000 Bytes

 Cluster Summary:
 ======================
 Cluster Total Capacity:   20219446689792 Bytes
```

See also

- “mmchlicense command” on page 239

Location

/usr/lpp/mmfs/bin

mmlsmgr command

Displays which node is the file system manager for the specified file systems or which node is the cluster manager.

Synopsis

```
mmlsmgr [Device [Device...]] [-Y]
```

or

```
mmlsmgr -C RemoteClusterName
```

or

```
mmlsmgr -c
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmlsmgr` command to display which node is the file system manager or cluster manager for the file system.

If you do not provide a *Device* operand, file system managers for all file systems within the current cluster for which a file system manager has been appointed are displayed.

Parameters

Device
The device names of the file systems for which the file system manager information is displayed.

If more than one file system is listed, the names must be delimited by a space. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

If no file system is specified, information about all file systems is displayed.

-C *RemoteClusterName*
Displays the name of the nodes that are file system managers in cluster *RemoteClusterName*.

-c
Displays the current cluster manager node.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

If you are a root user, the node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

As root, a user can also issue the `mmlsmgr` on remote file systems.

If you are a non-root user, you may specify only file systems that belong to the same cluster as the node on which the `mmlsmgr` command was issued.

Examples

1. To display the file system manager node information for all the file systems, issue this command:

   ```
   mmlsmgr
   ```

   The system displays information similar to:

   ```
   file system      manager node
   ---------------- ------------------
   fs3              9.114.94.65 (c154n01)
   fs2              9.114.94.73 (c154n09)
   fs1              9.114.94.81 (c155n01)

   Cluster manager node: 9.114.94.65 (c154n01)
   ```

   The output shows the device name of the file system and the file system manager's node number and name, in parenthesis, as they are recorded in the GPFS cluster data.

2. To display the file system manager information for file systems `gpfs2` and `gpfs3`, issue this command:

   ```
   mmlsmgr gpfs2 gpfs3
   ```

   The system displays information similar to:

   ```
   file system      manager node [from 199.116.68.69 (k156gn02)]
   ---------------- ------------------
   gpfs2              199.116.68.70 (k154gn02)
   gpfs3              199.116.68.72 (kolt2g_r1b42)
   ```

See also

- “mmchmgr command” on page 242

Location

`/usr/lpp/mmfs/bin`

mmlsmount command

Lists the nodes that have a given GPFS file system mounted.

Synopsis

```
mmlsmount {Device | all | all_local | all_remote | {-F DeviceFileName}} [-L]
            [-Y] [-C {all | all_remote | ClusterName[,ClusterName...]}]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmlsmount` command reports if a file system is in use at the time the command is issued. A file system is considered to be in use if it is explicitly mounted with the `mount` or `mmmount` command, or if it is mounted internally for the purposes of running some other GPFS command. For example, when you run the `mmrestripefs` command, the file system will be internally mounted for the duration of the command. If `mmlsmount` is issued in the interim, the file system will be reported as being in use by the `mmlsmount` command but, unless it is explicitly mounted, will not show up in the output of the `mount` or `df` commands.

Parameters

***Device* | all | all_local | all_remote | {-F *DeviceFileName*}**
Indicates the file system or file systems for which information is displayed.

Device
Indicates the device name of the file system for which information is displayed. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

all
Indicates all file systems known to this cluster.

all_local
Indicates all file systems owned by this cluster.

all_remote
Indicates all file systems owned by another cluster.

-F *DeviceFileName*
Specifies a file containing the device names, one per line, of the file systems for which information is displayed.

This must be the first parameter.

Options

-L
Specifies to list the nodes that have the file system mounted.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

-C {all | all_remote | *ClusterName*[,*ClusterName*...]}
Specifies the clusters for which mount information is requested. If one or more *ClusterName* is specified, only the names of nodes that belong to these clusters and have the file system mounted are displayed. The dot character ('.') can be used in place of the cluster name to denote the local cluster.

Option `-C all_remote` denotes all clusters other than the one from which the command was issued.

Option `-C all` refers to all clusters, local and remote, that can have the file system mounted. Option `-C all` is the default.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

If you are a root user, the node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

If you are a non-root user, you may specify only file systems that belong to the same cluster as the node on which the `mmlsmount` command was issued.

Examples

1. To see how many nodes have file system `fs2` mounted, issue this command:

```
mmlsmount fs2
```

The system displays output similar to:

```
File system fs2 is mounted on 3 nodes.
```

2. To display all mounted file systems:

```
mmlsmount all
```

The system displays output similar to:

```
File system fs1 is mounted on 17 nodes.
File system remotefs1 (remote.cluster:fs1) is mounted on 17 nodes.
```

3. To display all remotely mounted file systems:

```
mmlsmount all_remote
```

The system displays output similar to:

```
File system remotefs1 (remote.cluster:fs1) is mounted on 17 nodes.
```

4. To list the nodes having all file systems mounted:

```
mmlsmount all -L
```

The system displays output similar to:

```
File system fs1 is mounted on 3 nodes:
  192.168.105.32  c6f1c3vp2
  192.168.105.31  c6f1c3vp1
```

```
  192.168.105.34  c6f1c3vp4
File system gpfs1 is not mounted.
```

See also

- “mmmount command” on page 545
- “mmumount command” on page 732

Location

/usr/lpp/mmfs/bin

mmlsnodeclass command

Displays node classes defined in the system.

Synopsis

```
mmlsnodeclass [ClassName[,ClassName...] | --user | --system | --all] [-Y]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmlsnodeclass` command to display node classes defined in the system.

Parameters

ClassName
Displays the specified node class.

--user
Displays all user-defined node classes. This is the default.

--system
Displays all system-defined node classes.

--all
Displays both the system-defined and user-defined node classes.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmlsnodeclass` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To display the current user-defined node classes, issue this command:

```
mmlsnodeclass
```

The system displays information similar to:

```
Node Class Name       Members
--------------------- ------------------------------------------------------------
siteA                 linuxNodes
```

2. To display all node classes defined in the system, issue this command:

```
mmlsnodeclass --all
```

The system displays information similar to:

```
Node Class Name       Members
--------------------- ------------------------------------------------------------
aixNodes
all                   node03,node01,node02,node04,nodensd01,nodensd02
cesNodes              node01,node02,node04
clientLicense
clientNodes           node03,node02,node04,nodensd01,nodensd02
cnfsNodes
disabledCnfsNodes
enabledCnfsNodes
linuxNodes            node03,node01,node02,node04,nodensd01,nodensd02
managerNodes          node01
nonAixNodes           node03,node01,node02,node04,nodensd01,nodensd02
nonCesNodes           node03,nodensd01,nodensd02
nonCnfsNodes          node03,node01,node02,node04,nodensd01,nodensd02
nonLinuxNodes
nonNsdNodes           node003,node01,node02,node04
nonQuorumNodes
nonWindowsNodes       node03,node01,node02,node04,nodensd01,nodensd02
nsdNodes              nodensd01,nodensd02
quorumNodes           node03,node01,node02,node04,nodensd01,nodensd02
serverLicense         node03,node01,node02,node04,nodensd01,nodensd02
windowsNodes
siteA                 linuxNodes
object_database_node  node01
object_singleton_node node01
```

3. To display only the nodes that are quorum nodes, issue this command:

```
mmlsnodeclass quorumNodes
```

The system displays information similar to:

```
Node Class Name       Members
--------------------- ------------------------------------------------------------
quorumNodes           c8f2c1vp4,c8f2c4vp1,c8f2c4vp2
```

See also

- “mmchnodeclass command” on page 251
- “mmcrnodeclass command” on page 333
- “mmdelnodeclass command” on page 379

Location

/usr/lpp/mmfs/bin

mmlsnsd command

Displays Network Shared Disk (NSD) information for the GPFS cluster.

Synopsis

```
mmlsnsd [-a | -F | -f Device | -d "DiskName[;DiskName...]"]
        [-L | -m | -M | -X] [-Y  | -v]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmlsnsd` command to display the current information for the NSDs belonging to the GPFS cluster. The default is to display information for all NSDs defined to the cluster (`-a`). Otherwise, you may choose to display the information for a particular file system (`-f`) or for all disks that do not belong to any file system (`-F`).

Parameters

-a
Displays information for all of the NSDs belonging to the GPFS cluster. This is the default.

-f *Device*
Specifies the device name of the file system for which you want NSD information displayed. File system names need not be fully-qualified. `fs0` is as acceptable as `/dev/fs0`.

-F
Displays the NSDs that are not in use.

-d *DiskName*[;*DiskName*...]

Specifies the name of the NSDs for which you want information displayed. When you enter multiple *DiskName*s, separate them with semicolons and enclose the entire string of disk names in quotation marks:

```
"gpfs3nsd;gpfs4nsd;gpfs5nsd"
```

Options

-L
Displays the information in a long format that shows the NSD identifier.

-m
Maps the NSD name to its disk device name on the local node and, if applicable, on the NSD server nodes.

-M
Maps the NSD names to its disk device name on all nodes.

This is a slow operation and its usage is suggested for problem determination only.

-v
Specifies that the output should contain error information, where available.

-X
Maps the NSD name to its disk device name on the local node and, if applicable, on the NSD server nodes. The **-X** option also displays extended information for the NSD volume ID and information such as NSD server status and Persistent Reserve (PR) enablement in the Remarks field. Using the **-X** option is a slow operation and is recommended only for problem determination.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to issue the `mmlsnsd` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To display the default information for all of the NSDs belonging to the cluster, issue this command:

```
mmlsnsd
```

The system displays information similar to:

```
 File system   Disk name    NSD servers
------------------------------------------------------------------------
 fs2           hd3n97       c5n97g,c5n98g,c5n99g
 fs2           hd4n97       c5n97g,c5n98g,c5n99g
 fs2           hd5n98       c5n98g,c5n97g,c5n99g
 fs2           hd6n98       c5n98g,c5n97g,c5n99g
 fs2           hd7n97       c5n97g,c5n98g,c5n99g
 fs2           hd8n97       c5n97g,c5n98g,c5n99g
 fs2           hd9n97       c5n97g,c5n98g,c5n99g
 fs2           hd10n98      c5n98g,c5n97g,c5n99g
 fs2           hd11n98      c5n98g,c5n97g
 fs2           hd12n98      c5n98g,c5n97g
 fs2           sdbnsd       c5n94g,c5n96g
 fs2           sdcnsd       c5n94g,c5n96g
 fs2           sddnsd       c5n94g,c5n96g
 fs2           sdensd       c5n94g,c5n96g
 fs2           sdgnsd       c5n94g,c5n96g
 fs2           sdfnsd       c5n94g,c5n96g
 fs2           sdhnsd       c5n94g,c5n96g
 (free disk)   hd2n97       c5n97g,c5n98g
```

2. To display all of the NSDs attached to the node from which the command is issued, issue this command:

```
mmlsnsd -m
```

The system displays information similar to:

```
 Disk name    NSD volume ID      Device         Node name  Remarks
---------------------------------------------------------------------
 hd10n98      0972846245C8E93C   /dev/hd10n98   c5n97g     server node
 hd10n98      0972846245C8E93C   /dev/hd10n98   c5n98g     server node
 hd11n98      0972846245C8E93F   /dev/hd11n98   c5n97g     server node
 hd11n98      0972846245C8E93F   /dev/hd11n98   c5n98g     server node
 hd12n98      0972846245C8E941   /dev/hd12n98   c5n97g     server node
 hd12n98      0972846245C8E941   /dev/hd12n98   c5n98g     server node
```

```
 hd2n97        0972846145C8E924   /dev/hdisk2    c5n97g     server node
 hd2n97        0972846145C8E924   /dev/hdisk2    c5n98g     server node
 hd3n97        0972846145C8E927   /dev/hdisk3    c5n97g     server node
 hd3n97        0972846145C8E927   /dev/hdisk3    c5n98g     server node
 hd4n97        0972846145C8E92A   /dev/hdisk4    c5n97g     server node
 hd4n97        0972846145C8E92A   /dev/hdisk4    c5n98g     server node
 hd5n98        0972846245EB501C   /dev/hdisk5    c5n97g     server node
 hd5n98        0972846245EB501C   /dev/hdisk5    c5n98g     server node
 hd6n98        0972846245DB3AD8   /dev/hdisk6    c5n97g     server node
 hd6n98        0972846245DB3AD8   /dev/hdisk6    c5n98g     server node
 hd7n97        0972846145C8E934   /dev/hd7n97    c5n97g     server node
```

3. To display all of the NSDs in the GPFS cluster in extended format, issue this command:

```
mmlsnsd -L
```

The system displays information similar to:

```
 File system   Disk name    NSD volume ID      NSD servers
------------------------------------------------------------
 fs2           hd3n97       0972846145C8E927   c5n97g,c5n98g
 fs2           hd4n97       0972846145C8E92A   c5n97g,c5n98g
 fs2           hd5n98       0972846245EB501C   c5n98g,c5n97g
 fs2           hd6n98       0972846245DB3AD8   c5n98g,c5n97g
 fs2           sdbnsd       0972845E45C8E8ED   c5n94g,c5n96g
 fs2           sdcnsd       0972845E45C8E8F6   c5n94g,c5n96g
 fs2           sddnsd       0972845E45F83FDB   c5n94g,c5n96g
 fs2           sdensd       0972845E45C8E909   c5n94g,c5n96g
 fs2           sdgnsd       0972845E45C8E912   c5n94g,c5n96g
 fs2           sdfnsd       0972845E45F02E81   c5n94g,c5n96g
 fs2           sdhnsd       0972845E45C8E91C   c5n94g,c5n96g
 gpfs1         hd2n97       0972846145C8E924   c5n97g,c5n98g
```

4. To display extended disk information about disks hd3n97, sdfnsd, and hd5n98, issue this command:

```
mmlsnsd -X -d "hd3n97;sdfnsd;hd5n98"
```

The system displays information similar to:

```
 Disk name    NSD volume ID      Device          Devtype  Node name  Remarks
---------------------------------------------------------------------------------
 hd3n97       0972846145C8E927   /dev/hdisk3     hdisk    c5n97g     server node,pr=no
 hd3n97       0972846145C8E927   /dev/hdisk3     hdisk    c5n98g     server node,pr=no
 hd5n98       0972846245EB501C   /dev/hdisk5     hdisk    c5n97g     server node,pr=no
 hd5n98       0972846245EB501C   /dev/hdisk5     hdisk    c5n98g     server node,pr=no
 sdfnsd       0972845E45F02E81   /dev/sdf        generic  c5n94g     server node
 sdfnsd       0972845E45F02E81   /dev/sdm        generic  c5n96g     server node
```

5. The following shows the output of `mmlsnsd -X` with `mmchconfig usePersistentReserve=yes`.

```
 Disk name    NSD volume ID      Device          Devtype  Node name                Remarks
-------------------------------------------------------------------------------------------
------
 nsd0         947AC0A84F5FB55C   /dev/dm-0       dmm      c13c1apv12.gpfs.net      server
node,pr=yes
 nsd0         947AC0A84F5FB55C   /dev/dm-19      dmm      c13c1apv13.gpfs.net      server
node,pr=yes
 nsd0         947AC0A84F5FB55C   /dev/dm-6       dmm      c13c1apv14.gpfs.net      server
node,pr=yes
 nsd0         947AC0A84F5FB55C   /dev/dm-15      dmm      c13c1apv16.gpfs.net      server
node,pr=yes
 nsd1         947AC0A84F5FB564   /dev/dm-1       dmm      c13c1apv12.gpfs.net      server
node,pr=yes
 nsd1         947AC0A84F5FB564   /dev/dm-4       dmm      c13c1apv13.gpfs.net      server
node,pr=yes
 nsd1         947AC0A84F5FB564   /dev/dm-9       dmm      c13c1apv14.gpfs.net      server
node,pr=yes
 nsd1         947AC0A84F5FB564   /dev/dm-13      dmm      c13c1apv16.gpfs.net      server
node,pr=yes
```

See also

- “mmcrnsd command” on page 335
- “mmdelnsd command” on page 381

Location

/usr/lpp/mmfs/bin

mmlspolicy command

Displays policy information.

Synopsis

```
mmlspolicy Device [-L] [-Y]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmlspolicy` command displays policy information for a given file system. The information displayed includes:

- When the policy file was installed.
- The user who installed the policy file.
- The node on which the policy file was installed.
- The first line of the original policy file.

For information about GPFS policies and file placement, see *Information Lifecycle Management* in *IBM Spectrum Scale: Administration Guide*.

Parameters

Device
The device name of the file system for which policy information is to be displayed. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

-L
Displays the entire original policy file. If this flag is not specified, only the first line of the original policy file is displayed.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. This command displays basic information for the policy installed for file system `fs2`:

```
mmlspolicy fs2
```

   The system displays output similar to:

```
Policy for file system '/dev/fs2':
   Installed by root@c103rp12.gpfs.net on Tue Mar 30 15:06:20 2010.
   First line of policy 'policy' is:
/* This is the policy for the fs2 GPFS file system. */
```

2. This command displays extended information for the policy installed for file system `fs2`:

```
mmlspolicy fs2 -L
```

   The system displays output similar to:

```
/* This is the policy for the fs2 GPFS file system. */

/* File Placement Rules */
RULE SET POOL 'sp4' WHERE name like '%sp4%'
RULE SET POOL 'sp5' WHERE name like '%sp5%'
RULE 'default' SET POOL 'system'

/* Snapshot Placement Rules*/
RULE SET SNAP_POOL 'sp1' WHERE SNAP_NAME LIKE '%daily%'
RULE SET SNAP_POOL 'sp2'

/* Exclude Rule */
RULE 'Exclude root users files' EXCLUDE WHERE USER_ID = 0 AND
 name like '%org%'

/* Delete Rule */
RULE 'delete files' DELETE WHERE PATH_NAME like '%tmp%'

/* Migrate Rule */
RULE 'sp4.files' MIGRATE FROM POOL 'sp4' TO POOL 'sp5' WHERE
 name like '%sp4%'

/* End of Policy */
```

3. In this example, no policy file was installed for the specified file system:

```
mmlspolicy fs4 -L
```

   The system displays output similar to:

```
No policy file was installed for file system 'fs4'.
Data will be stored in pool 'system'.
```

See also

- “mmapplypolicy command” on page 80
- “mmchpolicy command” on page 258

Location

`/usr/lpp/mmfs/bin`

mmlspool command

Displays information about the known storage pools.

Synopsis

```
mmlspool Device {StoragePool[,StoragePool...] | all} [-L]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmlspool` command displays basic or detailed information about the storage pools in a file system.

Parameters

Device
Specifies the device name of the file system for which storage pool information is to be displayed. File system names do not need to be fully qualified; for example, `fs0` is as acceptable as `/dev/fs0`.

***StoragePool*[,*StoragePool*...]**
Specifies one or more storage pools for which information is to be displayed.

all
Displays information about all the storage pools in specified file system.

-L
Displays detailed information about each storage pool.

Exit status

0
Successful completion.

nonzero
A failure occurred.

Security

If you are a root user, the node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

If you are a non-root user, you may specify only file systems that belong to the same cluster as the node on which the `mmlspool` command was issued.

Examples

1. To show basic information about all storage pools in a file system, issue this command:

   ```
   mmlspool /dev/fst all
   ```

   The system displays information similar to:

   ```
   Name                    Id
   system                   0
   sataXXX              65537
   ```

2. To show more information, issue this command:

```
mmlspool fs1 p1 -L
```

The system displays information similar to this:

```
Pool:
  name               = p1
  poolID             = 65537
  blockSize          = 4 MB
  usage              = dataOnly
  maxDiskSize        = 497 GB
  layoutMap          = cluster
  allowWriteAffinity = no
  writeAffinityDepth = 0
  blockGroupFactor   = 1
```

See also

- “mmlsattr command” on page 487
- “mmlscallback command” on page 490
- “mmlscluster command” on page 492
- “mmlsconfig command” on page 495
- “mmlsdisk command” on page 497
- “mmlspolicy command” on page 526
- “mmlsquota command” on page 535
- “mmlssnapshot command” on page 540

See also the following *IBM Spectrum Scale RAID: Administration* topics:

- "mmlsrecoverygroup command"
- "mmlsvdisk command"

Location

`/usr/lpp/mmfs/bin`

mmlsqos command

Displays the I/O performance values of a file system, when you enable Quality of Service for I/O operations (QoS) with the mmchqos command.

Synopsis

```
mmlsqos Device

   [-Y | --fine-stats DisplayRange]
   [--pool {all | Pool}]
   [--seconds Seconds]
   [--sum-classes {yes | no}]
   [--sum-nodes {yes | no}]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Attention: The `mmqos` command provides the same functionality as both the `mmchqos` command and the `mmlsqos` command and has additional features. Future QoS features will be added to the `mmqos` command rather than to either the `mmchqos` command or the `mmlsqos` command. For more information, see "mmqos command" on page 619.

With the `mmlsqos` command, you can display the consumption of I/O operations by processes that access designated storage pools. With the `mmchqos` command, you can regulate I/O access to a specified storage pool by allocating shares of I/O operations to two QoS classes:

`maintenance`
The default QoS class for some I/O intensive, potentially long-running GPFS commands, such as `mmbackup`, `mmrestore`

`other`
The default QoS class for all other processes.

A third class, `misc`, is used to count the IOPS that some critical file system processes consume. You cannot assign IOPS to this class, but its count of IOPS is displayed in the output of the `mmlsqos` command.

Remember the following points:

- Allocations persist across unmounting and remounting the file system.
- QoS stops applying allocations when you unmount the file system and resumes when you remount it.
- When you change allocations or mount the file system, a brief delay due to reconfiguration occurs before QoS starts applying allocations.

For more information about this command, see the topic *Setting the Quality of Service for I/O operations (QoS)* in the *IBM Spectrum Scale: Administration Guide*.

When the file system is mounted, the command displays information about the QoS classes of both explicitly named pools and unnamed pools. *Unnamed pools* are storage pools that you have not specified by name in any `mmchqos` command. When the file system is unmounted, the command displays information about only the QoS classes of explicitly named pools.

Parameters

Device
The device name of the file system to which the QoS action applies.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

--fine-stats ***DisplayRange***
Displays the fine-grained statistics that are currently in memory. The *DisplayRange* values specify the indexes of the first and last blocks of statistics in the range of blocks that you want to display, such as `0-4`. If you specify only one integer, such as 2, the command displays all the blocks in memory starting with that index and going to the end. The last line of the output displays the index of the next block in memory. You can avoid re-displaying statistics by having the next `mmlsqos` command display statistics beginning at this block index.

Fine-grained statistics are taken at 1-second intervals and contain more information than regular statistics. They are intended to be used as input for programs that analyze and display data, such as the example plotting program at `/usr/lpp/mmfs/samples/charts/qosplotfine.pl`. For the content of the statistics, see the subtopic later in this topic.

--pool
Displays the I/O performance values for all QoS pools if `all` is specified, or for the named pool if a pool name is specified. The default is `all`.

--seconds
Displays the I/O performance values for the previous number of seconds. The valid range of seconds is 1-999. The default value is 60 seconds. The values are displayed for subperiods within the period that you specify. The subperiods might be every 5 seconds over the last 60 seconds, or every 60 seconds over the last 600 seconds. You cannot configure the number or length of subperiods.

--sum-classes
Displays the I/O performance for each QoS class separately if no is specified, or summed across all the QoS classes if `yes` is specified. The default is no.

--sum-nodes
If `yes` is specified, displays the I/O performance summed across all the nodes in the cluster. If no is specified, displays the I/O performance for each node separately. The default is `yes`.

Exit status

0
Successful completion.

Nonzero
A failure occurred.

Security

You must have root authority to run the `mmlsqos` command.

The node on which you enter the command must be able to execute remote shell commands on any other administration node in the cluster. It must be able to do so without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

Analyzing regular output from mmlsqos

The `mmlsqos` command always shows the first three lines of output:

QOS config::
Indicates whether QoS is actively regulating I/O consumption (enabled) or is quiescent (disabled).

QOS values::
Displays, for each storage pool that you configured, the name of the storage pool and the IOPS that you assigned to the `other` class and the `maintenance` class. In the following example fragment, the command shows that the system storage pool is configured with the value of `inf` for both QoS classes:

```
QOS values:: pool=system,other=inf,maintenance/all_local=inf
```

The qualifier `/all_local` after `maintenance` indicates that the maintenance IOPS are applied to all the files systems owned by the cluster. This value is the default for the `maintenance` class.

QOS status::
Indicates whether QoS is regulating the consumption of IOPS ("throttling") and also whether QoS is recording ("monitoring") the consumption of IOPS of each storage pool.

The following sample output is complete:

```
# mmlsqos fs --seconds 30
QOS config::     enabled
QOS values::     pool=system,other=inf,maintenance/all_local=inf:pool=fpodata,other=inf,maintenance/
all_local=inf
QOS status::     throttling active, monitoring active
=== for pool fpodata
01:31:45  misc iops=11 ioql=0.016539 qsdl=1.2e-06 et=5
=== for pool system
01:31:45  misc iops=8.2 ioql=0.013774 qsdl=2e-06 et=5
```

The command `mmlsqos fs0 --seconds 30` requests a display of I/O performance values for all QoS pools over the previous 30 seconds. Because the parameters `--sum_classes` and `--sum_nodes` are missing, the command also requests I/O performance for each storage pool separately and summed across all the nodes of the cluster.

The information that is displayed for the two configured pools, `fpodata` and `system`, indicates that IOPS occurred only for processes in the `misc` class. The meaning of the categories in each line is as follows:

First column
The time when the measurement period ends.

Second column
The QoS class for which the measurement is made.

iops=
The performance of the class in I/O operations per second.

ioql=
The average number of I/O requests in the class that are pending for reasons other than being queued by QoS. This number includes, for example, I/O requests that are waiting for network or storage device servicing.

qsdl=
The average number of I/O requests in the class that are queued by QoS. When the QoS system receives an I/O request from the file system, QoS first finds the class to which the I/O request belongs. It then finds whether the class has any I/O operations available for consumption. If not, then QoS queues the request until more I/O operations become available for the class. The `Qsdl` value is the average number of I/O requests that are held in this queue.

et=
The interval in seconds during which the measurement was made.

You can calculate the average service time for an I/O operation as *(Ioql + Qsdl)/Iops*. For a system that is running IO-intensive applications, you can interpret the value *(Ioql + Qsdl)* as the number of threads in the I/O-intensive applications. This interpretation assumes that each thread spends most of its time in waiting for an I/O operation to complete.

Analyzing fine-grained output from mmlsqos

The following code block shows an example of fine-grained output. Each line displays the sum of the data for all the QoS programs that are running on the specified node. If the `--pid-stats` parameter is

specified in the mmchqos command, then each line displays the data for one QoS program that is running on the specified node.

```
Time, Class, Node, Iops, TotSctrs, Pool, Pid, RW, SctrI, AvgTm, SsTm, MinTm, MaxTm, AvgQd, SsQd
1480606227,misc,172.20.0.21,285,2360,system,0,R,16,0.000260,0.000197,0.000006,0.005894,0.000001,0.000000
1480606228,misc,172.20.0.21,631,5048,system,0,R,16,0.000038,0.000034,0.000006,0.002463,0.000000,0.000000
1480606239,other,172.20.0.21,13,112,system,26724,R,16,0.000012,0.000000,0.000008,0.000022,0.000001,0.000000
1480606239,other,172.20.0.21,1,512,system,26724,R,512,0.000375,0.000000,0.000375,0.000375,0.000002,0.000000
1480606239,other,172.20.0.21,30,15360,system,26724,W,512,0.000910,0.000004,0.000451,0.002042,0.000001,0.000000
1480606241,misc,172.20.0.21,5,48,system,14680072,W,16,0.000135,0.000000,0.000025,0.000258,0.000000,0.000000
1480606240,other,172.20.0.21,4,32,system,26724,R,16,0.000017,0.000000,0.000009,0.000022,0.000001,0.000000
1480606240,other,172.20.0.21,48,24576,system,26724,W,512,0.000916,0.000006,0.000490,0.002141,0.000001,0.000000
1480606241,other,172.20.0.21,10,80,system,26724,R,16,0.000017,0.000000,0.000007,0.000039,0.000001,0.000000
1480606241,other,172.20.0.21,34,17408,system,26724,W,512,0.000873,0.000007,0.000312,0.001957,0.000001,0.000000
1480606241,misc,172.20.0.21,12,104,system,15597572,W,16,0.000042,0.000000,0.000024,0.000096,0.000000,0.000000
1480606241,misc,172.20.0.21,1,512,system,15597572,W,512,0.000192,0.000000,0.000192,0.000192,0.000000,0.000000
1480606241,misc,172.20.0.21,9,72,system,14680071,W,16,0.000029,0.000000,0.000024,0.000040,0.000000,0.000000
## conti=4
```

The following list describes the content in each column:

Time
The time when the measurement was made, expressed in seconds since the epoch.

Class
The QoS class of the process.

Node
The IP address of the node on which the process was running.

Iops
The number of IOPS that the process consumed during the sample period.

TotSctrs
The number of sectors for which data was read or written. By default, one sector is 512 bytes.

Note: The number of **TotSctrs** from one process or one node might not be equal to the real bytes read or written from applications. For cache I/O in IBM Spectrum Scale, if the modified data from one application I/O is less than **fgdbRangeSize** (by default, it is 4KB), IBM Spectrum Scale writes bytes defined by **fgdbRangeSize** from pagepool to backend disks. Therefore, if the application updates only a byte of the file, it shows eight sectors from the **mmlsqos** output when taking the default **fgdbRangeSize**.

Pool
The memory pool for which the IOPS were used.

Pid
The process ID of the process for which the IOPS were used. When the `--pid-stats` parameter of the mmchqos command is not specified, this value is always 0.

RW
Whether the process was doing a read operation or a write operation.

SctrI
The number of sectors for which the IOPS were done. One of the following categories is displayed. The exact value of a category depends on the disk block size.

A single sector.
1/32 or less of a full block.
Less than a full block.
A full block.

For example, if the disk block size is 512, the command displays one of the following values: 1, 16, 511, or 512.

AvgTm
The mean time that was required for an I/O operation to be completed.

SsTm
The sum of the squares of differences from the mean value that is displayed for AvgTm. You can use this value to calculate the variance of I/O service times.

MinTm
The minimum time that was required for an I/O operation to be completed.

MaxTm
The maximum time that was required for an I/O operation to be completed.

AvgQd
The mean time for which QoS imposed a delay of the read or write operation.

SsQd
The sum of the squares of differences from the mean value that is displayed for `AvgQd`.

The last line in the example indicates that the index of the next block to be displayed is 4. You can avoid re-displaying statistics by having the next `mmlsqos` command display statistics beginning at this block index.

Examples

1. The following command displays the I/O performance values for all the pools in the file system over the previous 60 seconds. It does so for each QoS class separately and summed across all the nodes in the cluster.

```
mmlsqos fs0 --seconds 60
```

2. The following command displays the I/O performance values for the named pool over the previous 60 seconds. It does so for each QoS class separately and for each node separately.

```
mmlsqos fs0 --pool pname0 --sum-nodes no
```

See also

- “mmchqos command” on page 263
- *Setting the Quality of Service for I/O operations (QoS)* in the *IBM Spectrum Scale: Administration Guide*.

Location

`/usr/lpp/mmfs/bin`

mmlsquota command

Displays quota information for a user, group, or fileset.

Synopsis

```
mmlsquota [-u User | -g Group] [-v | -q] [-e] [-C ClusterName]
          [-Y] [--block-size {BlockSize | auto}] [Device[:Fileset] ...]
```

or

```
mmlsquota -j Fileset [-v | -q] [-e] [-C ClusterName]
          [-Y] [--block-size {BlockSize | auto}] Device ...
```

or

```
mmlsquota -d {[-u] [-g] [-j]} [-C ClusterName]
          [-Y] [--block-size {BlockSize | auto}] [Device ...]
```

or

```
mmlsquota -d [-C ClusterName] [-Y] [--block-size {BlockSize | auto}] [Device[:Fileset] ...]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

For the specified *User*, *Group*, or *Fileset* the `mmlsquota` command displays information about quota limits and current usage on each file system in the cluster. This information is displayed only if quota limits have been established and the user has consumed some amount of storage. If you want quota information for a *User*, *Group*, or *Fileset* that has no file system storage allocated at the present time, you must specify `-v`.

Important: Quota limits are not enforced for root users (by default). For information on managing quotas, see *Managing GPFS quotas* in the *IBM Spectrum Scale: Administration Guide*.

If neither the `-g`, `-u`, or `-j` option is specified, the default is to display only user quotas for the user who issues the command.

Replication:

- Replicated data is included in data block current usage. See the note under "Current usage" in the "Block limits" list later in this topic.
- Data replication and metadata replication do not affect the current number of files that are reported for quotas. See the note under "Current number of files" in the "File limits" list later in this topic.

For each file system in the cluster, the `mmlsquota` command displays:

1. Block limits:
   - Quota type (USR or GRP or FILESET)
   - Current usage

     Note: If data replication is enabled, the data block current usage includes the replicated data. For more information, see the topic *Listing quotas* in the *IBM Spectrum Scale: Administration Guide*.
   - Soft limit
   - Hard limit
   - Space in doubt
   - Grace period

2. File limits:
   - Current number of files

     Note: The current number of files is not increased by data replication or metadata replication. For more information, see the topic *Listing quotas* in the *IBM Spectrum Scale: Administration Guide*.
   - Soft limit
   - Hard limit
   - Files in doubt
   - Grace period

   Note:
   - In cases where small files do not have an additional block allocated for them, quota usage might show less space usage than expected.
   - If you want to check the grace period that is set, specify `mmrepquota -t`.

Because the sum of the *in-doubt* value and the current usage may not exceed the hard limit, the actual block space and number of files available to the user, group, or fileset may be constrained by the in-doubt value. If the in-doubt value approaches a significant percentage of the quota, run the `mmcheckquota` command to account for the lost space and files. For more information, see *Listing quotas* in *IBM Spectrum Scale: Administration Guide*.

This command cannot be run from a Windows node.

Parameters

-C *ClusterName*
Specifies the name of the cluster from which the quota information is obtained (from the file systems within that cluster). If `-C` is omitted, the local cluster is assumed. The cluster name specified by the `-C` flag must be part of the same multicluster group as the node issuing the `mmlsquota` command. A node that is part of a remote cluster can only see the file systems that it has been given authority to mount from the local cluster.

Device
Specifies the device name of the file system for which quota information is to be displayed. File system names need not be fully-qualified. `fs0` is as acceptable as `/dev/fs0`.

Fileset
Specifies the name of a fileset located on *Device* for which quota information is to be displayed.

-d
Displays the default quota limits for user, group, or fileset quotas. When specified in combination with the `-u`, `-g`, or `-j` options, default file system quotas are displayed. When specified without any of the `-u`, `-g`, or `-j` options, default fileset-level quotas are displayed.

-e
Specifies that `mmlsquota` is to collect updated quota usage data from all nodes before displaying results. If `-e` is not specified, there is the potential to display negative usage values as the quota server may process a combination of up-to-date and back-level information.

-g *Group*
Displays quota information for the user group or group ID specified in the *Group* parameter.

-j *Fileset*
Displays quota information for the named fileset.

-q
Prints a terse message containing information only about file systems with usage over quota.

-u *User*
Displays quota information for the user name or user ID specified in the *User* parameter.

-v
Displays quota information on file systems where the *User*, *Group* or *Fileset* limit has been set, but the storage has not been allocated.

--block-size {*BlockSize* | auto}
Specifies the unit in which the number of blocks is displayed. The value must be of the form [*n*]K, [*n*]M, [*n*]G or [*n*]T, where *n* is an optional integer in the range 1 to 1023. The default is 1K. If `auto` is specified, the number of blocks is automatically scaled to an easy-to-read value.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note:

- Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.
- -Y disregards the value of --block-size and returns the results in KB.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

If you are a root user:

- You may view quota information for all users, groups, and filesets.
- The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

If you are a non-root user, you may view only fileset quota information, your own quota information, and quota information for any groups to which you belong.

You must be a root user to use the `-d` option.

GPFS must be running on the node from which the `mmlsquota` command is issued.

Examples

1. The user ID `paul` issues this command:

```
mmlsquota
```

The system displays information similar to:

```
                         Block Limits            |     File Limits
Filesystem type KB  quota  limit in_doubt grace |files quota limit in_doubt grace
fsn        USR 728 100096 200192  4880     none |   35    30    50    10
6days
```

This output shows the quotas for user `paul` in file system `fsn` set to a soft limit of 100096 KB, and a hard limit of 200192 KB. 728 KB is currently allocated to `paul`. 4880 KB is also in doubt, meaning that the quota system has not yet been updated as to whether this space has been used by the nodes, or whether it is still available. No grace period appears because `paul` has not exceeded his quota. If he exceeds the soft limit, the grace period is set and he has that amount of time to bring his usage below the quota values. If he fails to do so, he can not allocate any more space.

The soft limit for files (inodes) is set at 30 and the hard limit is 50. 35 files are currently allocated to this user, and the quota system does not yet know whether the 10 in doubt have been used or are still available. A grace period of six days appears because the user has exceeded his quota. The user would have this amount of time to bring his usage below the quota values. If the user fails to do so, the user is not allocated any more space.

2. To show the quotas for user pfs001, device gpfs2, and fileset fset4, issue this command:

```
mmlsquota -u pfs001 gpfs2:fset4
```

The system displays information similar to:

```
                        Block Limits                    |     File Limits
                                              in                         in
Filesystem Fileset type   KB quota  limit doubt grace | files quota limit doubt grace
Remarks
gpfs2      fset4   USR  4104 10240 153600     0  none |     1  1000  5000     0  none
```

3. To show user and group default quotas for all filesets in the gpfs1 file system, issue this command:

```
mmlsquota -d gpfs1
```

The system displays information similar to:

```
         Default Block Limits(KB)                 |  Default File Limits
Filesystem Fileset    type      quota      limit  |     quota    limit  entryType
gpfs1      root       USR           0          0  |         0        0  default off
gpfs1      root       GRP           0          0  |         0        0  default off
gpfs1      fset1      USR           0          0  |         0        0  default off
gpfs1      fset1      GRP           0          0  |         0        0  default off
gpfs1      fset2      USR           0          0  |         0        0  default off
gpfs1      fset2      GRP           0          0  |         0        0  default off
```

4. To show user and group default quotas for fileset fset1 in the gpfs1 file system, issue this command:

```
mmlsquota -d gpfs1:fset1
```

The system displays information similar to:

```
         Default Block Limits(KB)       |  Default File Limits
Filesystem Fileset    type      quota      limit  |     quota    limit  entryType
gpfs1      fset1      USR           0          0  |         0        0  default off
gpfs1      fset1      GRP           0          0  |         0        0  default off
```

5. To show the quotas for fileset fset0 in file system fs1, issue this command:

```
mmlsquota -j fset0 fs1 --block-size auto
```

The system displays information similar to:

```
                     Block Limits                                |     File Limits
Filesystem type      blocks   quota   limit   in_doubt    grace |  files   quota    limit  in_doubt
grace  Remarks
fs1        FILESET   89.25G    100G    200G     89.99M     none |  13729    4000     5000         0  7
days
```

See also

- “mmcheckquota command” on page 220
- “mmdefedquota command” on page 345
- “mmdefquotaoff command” on page 349
- “mmdefquotaon command” on page 352
- “mmedquota command” on page 404
- “mmrepquota command” on page 666

- “mmquotaon command” on page 654
- “mmquotaoff command” on page 651

Location

/usr/lpp/mmfs/bin

mmlssnapshot command

Displays GPFS snapshot information.

Synopsis

```
mmlssnapshot Device [-d [--fast][--block-size {BlockSize | auto}]]
    [-s {all | global | [[Fileset]:]Snapshot[,[[Fileset]:]Snapshot...]} | -j
Fileset[,Fileset...]]
    [--qos QOSClass] [-Y]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmlssnapshot` command to display GPFS snapshot information for the specified file system or fileset. You can optionally display the amount of storage that is used by the snapshot.

Parameters

Device
The device name of the file system for which snapshot information is to be displayed. File system names do not need to be fully qualified. `fs0` is as acceptable as `/dev/fs0`.

-d
Displays the amount of storage that is used by the snapshot.

This operation requires an amount of time that is proportional to the size of the file system; therefore, it can take several minutes or even hours on a large and heavily-loaded file system.

This optional parameter can impact overall system performance. Avoid running the `mmlssnapshot` command with this parameter frequently or during periods of high file system activity.

--fast
Enables a faster way of calculating the amount of storage that is used by the snapshot, minimizing the impact on the performance of the system. This is achieved by getting the required details from the inode of the snapshots without locking the snapshot files, and by reducing the scanning on the indirect block tree of snapshot files. This results in the reduced I/O load as well. However, without proper locks on the snapshots, the calculation of metadata storage space for the latest snapshot or the previous snapshot that is being emptied might be inaccurate. The **`--fast`** option is available only if the minimum release level of the IBM Spectrum Scale cluster is upgraded to 5.1.1 or later.

--block-size {*BlockSize* | auto}
Specifies the unit in which the number of blocks is displayed. The value must be of the form [*n*]K, [*n*]M, [*n*]G or [*n*]T, where *n* is an optional integer in the range 1 - 1023. The default is 1 K. If `auto` is specified, the number of blocks is automatically scaled to an easy-to-read value.

-s
Displays the attributes for the specified snapshots.

all
Displays information for all snapshots. This option is the default.

global
Displays information for global snapshots.

[[*Fileset*]:]

:

A colon (:) followed by a snapshot name indicates a global snapshot. For example, `:SS01` indicates a global snapshot with the name SS01. If a global snapshot with that name exists, the command displays information about it.

Fileset:
A fileset name followed by a colon (:) followed by a snapshot name indicates a fileset snapshot. For example, `fset02:SS01` indicates a snapshot of fileset `fset02` with the name SS01. If a snapshot of the fileset with that snapshot name exists, then the command displays information about it.

***Snapshot*[,*Snapshot*...]**
Displays information for the specified snapshots.

-j *Fileset*[,*Fileset*...]
Displays only snapshots that contain the specified filesets; including all global snapshots.

--qos *QosClass*
Specifies the Quality of Service for I/O operations (QoS) class to which the instance of the command is assigned. If you do not specify this parameter, the instance of the command is assigned by default to the `maintenance` QoS class. This parameter has no effect unless the QoS service is enabled. For more information, see the topic "mmchqos command" on page 263. Specify one of the following QoS classes:

`maintenance`
This QoS class is typically configured to have a smaller share of file system IOPS. Use this class for I/O-intensive, potentially long-running GPFS commands, so that they contribute less to reducing overall file system performance.

other
This QoS class is typically configured to have a larger share of file system IOPS. Use this class for administration commands that are not I/O-intensive.

For more information, see the topic *Setting the Quality of Service for I/O operations (QoS)* in *IBM Spectrum Scale: Administration Guide*.

-Y
Displays the command output in a parseable format with a colon (:) as the field delimiter. Each column is described by a header.

Note: Fields having a colon are encoded to prevent confusion. If a field contains a % (percent sign) character, it is most likely encoded. Use the **`mmclidecode`** command to decode the field.

Exit status

0
Successful completion.

nonzero
A failure occurred.

Security

You must be a root user or fileset owner to use the `-d` parameter.

If you are a root user, the node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

If you are a non-root user, you can specify only file systems that belong to the same cluster as the node on which the `mmlssnapshot` command was issued.

Examples

Note: Ensure that the snapshot name does not include a colon (:).

The following command displays information about all the existing snapshots in file system `fs1`:

```
mmlssnapshot fs1
```

The following command displays information about snapshots named SS01. The snapshots can be global snapshots or fileset snapshots:

```
mmlssnapshot fs1 -s SS01
```

The following command displays information about a global snapshot named gSS01:

```
mmlssnapshot fs1 -s :gSS01
```

The following command displays information about a fileset snapshot with the name `fsSS01` that is a snapshot of fileset `fset02`:

```
mmlssnapshot fs1 -s fset02:fsSS01
```

The following command displays information about global snapshots with the names gSS01 and gSS02 and a fileset snapshot of fileset `fset02` named `fsSS02`:

```
mmlssnapshot fs1 -s :gSS01,:gSS02,fset02:fsSS02
```

The following command displays information about global snapshots and fileset snapshots that contain the filesets `fset02` and `fset03`:

```
mmlssnapshot fs1 -j fset02,fset03
```

See also

- “mmcrsnapshot command” on page 340
- “mmdelsnapshot command” on page 383
- “mmrestorefs command” on page 675
- “mmsnapdir command” on page 722

Location

`/usr/lpp/mmfs/bin`

mmmigratefs command

Performs needed conversions to support new file system features.

Synopsis

```
mmmigratefs Device [--fastea] [--online | --offline]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmmigratefs` command to enable features that require existing on-disk data structures to be converted to a new format.

Before issuing the `mmmigratefs` command, see upgrade, coexistence, and compatibility considerations in *Upgrading* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*. You must ensure that all nodes in the cluster have been upgraded to the latest level of GPFS code and that you have successfully run the `mmchconfig release=LATEST` command. You must also ensure that the new features have been enabled by running `mmchfs -V full`.

The `mmmigratefs` command can be run with the file system mounted or unmounted. If `mmmigratefs` is run without the `--online` or `--offline` parameters specified, the command will determine the mount status of the file system and run in the appropriate mode.

Parameters

Device
The device name of the file system to be migrated. File system names need not be fully qualified; for example, `fs0` is just as acceptable as `/dev/fs0`. This must be the first parameter.

--fastea
Convert the existing extended attributes to the new format required for storing the attributes in the file's inode and thereby allowing for faster extended-attribute access.

--online
Allows the `mmmigratefs` command to run while the file system is mounted.

--offline
Allows the `mmmigratefs` command to run while the file system is unmounted.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmmigratefs` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

To enable fast extended attribute access for file system fs3, issue this command:

```
mmmigratefs fs3 --fastea
```

The system displays information similar to the following:

```
Enabling fastea support
  11.19 % complete on Thu Nov 14 13:50:24 2013  (    167936 inodes        328 MB)
  33.21 % complete on Thu Nov 14 13:51:56 2013  (    498260 inodes        973 MB)
 100.00 % complete on Thu Nov 14 13:52:04 2013
Finalizing upgrade
  11.19 % complete on Thu Nov 14 13:52:27 2013  (    167936 inodes        328 MB)
  26.78 % complete on Thu Nov 14 13:53:07 2013  (    401834 inodes        785 MB)
 100.00 % complete on Thu Nov 14 13:53:27 2013
Feature 'fastea' is now enabled on "fs3".
```

See also

- “mmchconfig command” on page 170
- “mmchfs command” on page 232

Location

/usr/lpp/mmfs/bin

mmmount command

Mounts GPFS file systems on one or more nodes in the cluster.

Synopsis

```
mmmount {Device | DefaultMountPoint | DefaultDriveLetter |
         all | all_local | all_remote | {-F DeviceFileName}}
        [-o MountOptions] [-a | -N {Node[,Node...] | NodeFile | NodeClass}]
```

or

```
mmmount Device {MountPoint | DriveLetter}
        [-o MountOptions] [-a | -N {Node[,Node...] | NodeFile | NodeClass}]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmmount` command mounts the specified GPFS file system on one or more nodes in the cluster. If no nodes are specified, the file systems are mounted only on the node from which the command was issued. A file system can be specified using its device name or its default mount point, as established by the `mmcrfs`, `mmchfs` or `mmremotefs` commands.

When `all` is specified in place of a file system name, all GPFS file systems will be mounted. This also includes remote GPFS file systems to which this cluster has access.

Important: The `mmmount` command fails if you run it while file system maintenance mode is enabled. For more information about file system maintenance mode, see the topic *File system maintenance mode* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Note: The time that is required to mount a file system for the first time is greater if the file system uses one or more thin provisioned disks. The extra time is used by the mount operation to reserve physical space for error recovery on each thin provisioned disk. For more information, see the topic *IBM Spectrum Scale with data reduction storage devices* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Parameters

***Device* | *DefaultMountPoint* | *DefaultDriveLetter* | all | all_local | all_remote | {-F *DeviceFileName*}**
Indicates the file system or file systems to be mounted.

Device
The device name of the file system to be mounted. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

DefaultMountPoint
The mount point associated with the file system as a result of the `mmcrfs`, `mmchfs`, or `mmremotefs` commands.

DefaultDriveLetter
The Windows drive letter associated with the file system as a result of the `mmcrfs` or `mmchfs` command.

all
Indicates all file systems known to this cluster.

all_local
Indicates all file systems owned by this cluster.

all_remote
Indicates all files systems owned by another cluster to which this cluster has access.

-F *DeviceFileName*
Specifies a file containing the device names, one per line, of the file systems to be mounted.

This must be the first parameter.

DriveLetter
The location where the file system is to be mounted. If not specified, the file system is mounted at its default drive letter. This option can be used to mount a file system at a drive letter other than its default one or to mount a file system that does not have an established default drive letter.

MountPoint
The location where the file system is to be mounted. If not specified, the file system is mounted at its default mount point. This option can be used to mount a file system at a mount point other than its default mount point.

Options

-a
Mount the file system on all nodes in the GPFS cluster.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies the nodes on which the file system is to be mounted.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

This command does not support a *NodeClass* of `mount`.

-o *MountOptions*
Specifies the mount options to pass to the mount command when mounting the file system. For a detailed description of the available mount options, see *Mount options specific to IBM Spectrum Scale* in *IBM Spectrum Scale: Administration Guide*.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmmount` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see the topic *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

Examples

1. To mount all GPFS file systems on all of the nodes in the cluster, issue this command:

```
mmmount all -a
```

The output is similar to this:

```
Mon Mar 23 06:56:53 EDT 2020: mmmount: Mounting file systems ...
```

2. To mount file system `fs2` read-only on the local node, issue this command:

```
mmmount fs2 -o ro
```

   The output is similar to this:

```
Mon Mar 23 08:14:49 EDT 2020: mmmount: Mounting file systems ...
```

3. To mount file system `fs1` on all NSD server nodes, issue this command:

```
mmmount fs1 -N nsdsnodes
```

   The output is similar to this:

```
Mon Mar 23 08:18:53 EDT 2020: mmmount: Mounting file systems ...
```

See also

- “mmumount command” on page 732
- “mmlsmount command” on page 517

Location

`/usr/lpp/mmfs/bin`

mmnetverify command

Verifies network configuration and operation in a cluster.

Synopsis

```
mmnetverify [Operation[ Operation...]] [-N {Node[,Node...] | all}]
            [--target-nodes {Node[,Node...] | all}]
            [--configuration-file File] [--log-file File]
            [--verbose | -Y] [--min-bandwidth Number]
            [--max-threads Number] [--ces-override] [--ping-packet-size Number]
            [--subnets Addr[,Addr...]][--cluster Name Node[,Node...]]
```

or

```
mmnetverify --remote-cluster-daemon [--remote-cluster-port PortNumber]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Note: The `mmnetverify` command is a diagnostic tool and is not intended to be issued continually during normal IBM Spectrum Scale cluster operations. Issue this command only to validate a cluster network before going into production or when you suspect network problems.

With the `mmnetverify` command, you can verify the network configuration and operation of a group of nodes before you organize them into an IBM Spectrum Scale cluster. You can also run the command to analyze network problems after you create a cluster.

If you have not created an IBM Spectrum Scale cluster yet, you must run the command with a configuration file. See the `--configuration-file` option in the `Parameters` section.

The command uses the concepts of local nodes and target nodes. A *local node* is a node from which a network test is run. The command can be started from one node and run from several separate local nodes. A *target node* is a node against which a test is run.

The command has the following requirements:

- IBM Spectrum Scale must be installed on all the nodes that are involved in the test, including both local nodes and target nodes.
- Each node must be able to issue remote shell commands to all nodes, including itself, without a password. (This requirement is tested in the `shell` check.)

The following table lists the types of output messages and where they are sent. Messages are not added to a log file unless you specify the log-file name on the command line:

Table 25. Information and error messages

Message type	Printed to console	Added to the log file, if one is specified.
Information messages	Yes (`stdout`)	Yes
Verbose information messages	Yes (`stdout`) If `--verbose` is specified on the command line.	Yes
Error messages	Yes (`stderr`)	Yes

If sudo wrappers are enabled, the command uses a sudo wrapper when it communicates with remote nodes. Note the following restrictions:

- If you are working with an existing cluster, the cluster must be at IBM Spectrum Scale 4.2.3 or later.
- You must run the command on an administration node.
- The `-N` option is not supported.
- You must run the sudo command as the `gpfsadmin` user.

Parameters

***Operation*[Operation...]**
Specifies one or more operations, which are separated by blanks, that are to be verified against the target nodes. The operations are described in Table 27 on page 554. Shortcut terms are described in Table 26 on page 553.

If you do not specify any operations, the command does all the operations except `data-large`, `flood-node`, and `flood-cluster` against the target nodes.

-N {*Node*[,*Node*...] | all}
Specifies a list of nodes on which to run the command. If you specify more than one node, the command is run on all the specified nodes in parallel. Each node tests all the specified target nodes. If you do not include this parameter, the command runs the operations only from the node where you enter the command.

Note: The `--max-threads` parameter specifies how many nodes can run in parallel at the same time. If the limit is exceeded, the command still tests all the specified nodes, starting the surplus nodes as other nodes finish.

***Node*[,*Node*...]**
Specifies a list of nodes in the local cluster. This parameter accepts node classes. You can specify system node classes, such as **aixnodes**, and node classes that you define with the `mmcrnodeclass` command.

all
Specifies all the nodes in the local cluster.

--target-nodes {*Node*[,*Node*...] | all}]
Specifies a list of nodes that are the targets of the testing. If you do not specify this parameter, the command runs the operations against all the nodes in the cluster.

***Node*[,*Node*...]**
Specifies a list of nodes in the local cluster. This parameter accepts node classes. You can specify system node classes, such as **aixnodes**, and node classes that you define with the `mmcrnodeclass` command.

all
Specifies all the nodes in the local cluster.

[--configuration-file *File*]
Specifies the path of a configuration file. You must use a configuration file if you have not created an IBM Spectrum Scale cluster yet. You can also specify a configuration file if you have created a cluster but you do not want the command to run with the IBM Spectrum Scale cluster configuration values. Only the node parameter is required. The other parameters revert to their default values if they are not specified.

Note: When you specify this parameter, the `--ces-override` parameter is automatically set.

The following code block shows the format of the file:

```
node Node [AdminName]
rshPath Path
rcpPath Path
tscTcpPort Port
mmsdrservPort Port
```

```
tscCmdPortRange Min-Max
subnets Addr[,Addr...]
cluster Name Node[,Node...]
```

where:

node *Node* [*AdminName*]
Specifies a node name, followed optionally by the node's admin name. If you do not specify an admin name, the command uses the node name as the admin name.

You can have multiple node parameters. Add a node parameter for each node that you want to be included in the testing, either as a local node or as a target node. You must include the node from which you are running the command.

rshPath *Path*
Optional. Specifies the path of the remote shell command to be used. The default value is `/usr/bin/ssh`. Specify this parameter only if you want to use a different remote shell command.

rcpPath *Path*
Optional. Specifies the path of the remote file copy command to be used. The default value is `/usr/bin/scp`. Specify this parameter only if you want to use a different remote copy command.

tscTcpPort *Port*
Optional. Specifies the TCP port number to be used by the local GPFS daemon when it contacts a remote cluster. The default value is 1191. Specify this value only if you want to use a different port.

mmsdrservPort *Port*
Optional. Specifies the TCP port number to be used by the `mmsdrserv` service to provide access to configuration data to the rest of the nodes in the cluster. The default value is the value that is stored in `mmfsdPort`.

tscCmdPortRange=*Min-Max*
Specifies the range of port numbers to be used for extra TCP/IP ports that some administration commands need for their processing. Defining a port range makes it easier for you set firewall rules that allow incoming traffic on only those ports. For more information, see the topic *IBM Spectrum Scale port usage* in the *IBM Spectrum Scale: Administration Guide*.

If you used the **spectrumscale** installation toolkit to install a version of IBM Spectrum Scale that is earlier than version 5.0.0, then this attribute is initialized to 60000-61000. Otherwise, this attribute is initially undefined and the port numbers are dynamically assigned from the range of ephemeral ports that are provided by the operating system.

subnets *Addr*[,*Addr*...]
Optional. Specifies a list of subnet addresses to be searched for a subnet that both the local node and the target node are connected to. If such a subnet is found, the command runs the specified network check between the connections of the local node and target node to that subnet. Otherwise, the command runs the network check across another connection that the local node and the target node have in common. The command goes through this process for each local node and target node that are specified for the command to process. The default value of this parameter is no subnets.

You must specify subnet addresses in dot-decimal format, such as `10.168.0.0`. You cannot specify a cluster name as part of a subnet address. For more information about specifying a list of subnets, see the description of the parameter `--subnets` later in this topic.

cluster *Name Node*[,*Node*...]
Optional. Specifies the name of a remote cluster followed by at least one contact node identifier. The `remote-cluster` operation includes this cluster in its connectivity checks. You can specify this parameter multiple times in a configuration file. For more information, see the description of the `--cluster` parameter later in this topic.

[--log-file ***File*****]**
Specifies the path of a file to contain the output messages from the network checks. If you do not specify this parameter, messages are displayed only on the console. See Table 25 on page 548.

--verbose
Causes the command to generate verbose output messages. See Table 25 on page 548.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

[--min-bandwidth ***Number*****]**
Specifies the minimum acceptable bandwidth for the data bandwidth check.

--max-threads ***Number***
Specifies how many nodes can run the command in parallel at the same time. The valid range is 1 - 64. The default value is 32. For more information, see the `-N` parameter.

--ces-override
Causes the command to consider all the nodes in the configuration to be CES-enabled. This parameter overrides the requirement of the `protocol-ctdb` and the `protocol-object` network checks that the local node and the target nodes must be CES-enabled with the `mmchnode` command. This parameter is automatically set when you specify the `--configuration-file` parameter.

[--ping-packet-size ***Number*****]**
Specifies the size in bytes of the ICMP echo request packets that are sent between the local node and the target node during the **ping** test. The size must not be greater than the MTU of the network interface.

If the MTU size of the network interface changes, for example to support jumbo frames, you can specify this parameter to verify that all the nodes in the cluster can handle the new MTU size.

[--subnets ***Addr*****[,*****Addr*****...]]**
Specifies a list of subnets that the command searches in sequential order for a subnet that both the local node and the target node are connected to. If such a subnet is found, the command runs the network check between the connections of the local node and target node to that subnet. (The command runs the network check only for the first such subnet that it finds in the list.) If such a subnet is not found, the command runs the network check across another connection that the local node and the target node have in common. The command goes through this process for each local node and target node that are specified on the command line.

This parameter affects only the network checks that are included in the `port`, `data`, and `bandwidth` shortcuts. For a list of these network checks, see Table 26 on page 553 following.

Before you use this parameter, ensure that all nodes in the cluster or group are running IBM Spectrum Scale 5.0.0 or later and that you have successfully run the command `mmchconfig release=LATEST` on all the nodes. For more information, see the chapter *Migration, coexistence, and compatibility* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Addr**[,*****Addr*****...]**
Specifies a list of subnet addresses to be searched. You can specify a subnet address either as a literal network address in dot-decimal format, such as `10.168.0.0`, or as a shell-style regular expression that can match multiple subnet addresses.

Note: You cannot specify a cluster name as part of the subnet address.

The following types of regular expression are supported:

- Character classes

  - A set of numerals enclosed in square brackets. For example, the expression 192.168.[23].0 matches 192.168.2.0 and 192.168.3.0.
  - A range of numerals enclose in square brackets. For example, the expression 192.168.[2-16].0 matches the range 192.168.2.0 through 192.168.16.0.
- Quantifiers
  - The pattern X* signifies X followed by 0 or more characters. For example, the expression 192.168.*.0 matches 192.168.0.0, 192.168.1.0, and so on up to 192.168.255.0.
  - The pattern X? signifies X followed by 0 or 1 characters. For example, the expression 192.168.?.0 matches 192.168.0.0, 192.168.1, 0, and so on up to 192.168.9.0.

Tip: In all of IBM Spectrum Scale, you can specify a list of subnets for the `mmnetverify` command in three locations:

- In the `subnets` attribute of the `mmchconfig` command. For more information, see "mmchconfig command" on page 170.
- In the `subnets` entry of the `mmnetverify` configuration file. See the description of the `--configuration-file` parameter earlier in this topic.
- In the `--subnets` parameter of the `mmnetverify` command.

If you specify a list of subnets in the second location (the `subnets` entry of the `mmnetverify` configuration file) the command ignores any subnets that are specified in the first location. Similarly, if you specify a list of subnets in the third location (the `--subnets` parameter of the `mmnetverify` command) the command ignores any subnets that are specified in the first two locations.

[--cluster *Name Node*[,*Node*...]]
Specifies a remote cluster for the `remote-cluster` operation to check. This parameter can occur multiple times on the command line, so that you can specify multiple remote clusters. By default the `remote-cluster` operation checks only the known remote clusters -- that is, the remote clusters that are listed in the `mmsdrfs` file, where they are put by the `mmremotecluster` command. With the `--cluster` parameter, you can specify other remote clusters to be checked. For more information about the `remote-cluster` operation, see the entry for the operation in Table 27 on page 554 later in this topic.

***Node*[,*Node*...]]**
Specifies one or more contact nodes through which the **remote-cluster** operation can get information about the remote cluster. You must specify at least one contact node.

The `remote-cluster` operation evaluates a remote cluster in two stages. In the first stage it gathers information about the nodes in the remote cluster. In the second stage it checks the nodes for connectivity. The first stage has two phases:

1. In the first phase the `remote-cluster` operation tries to get information through the contact nodes of the remote cluster. If this attempt succeeds, the first stage ends and the operation goes to the second stage, which is testing the remote nodes for connectivity. But if the attempt fails, the operation goes to the second phase.
2. In the second phase the operation tries to get information through a special daemon called a *remote-cluster daemon* that can be running on a remote contact node. If this attempt succeeds, the first stage ends and the operation goes to the second stage, which is checking the nodes for connectivity. If the attempt fails, the operation displays an error message with instructions to start a remote-cluster daemon on a contact node and try the `remote-cluster` operation again.

To start a remote-cluster daemon, open a console window on a contact node and issue the following command:

```
mmnetverify --remote-cluster-daemon --remote-cluster-port <PortNumber>
```

Note: The `--remote-cluster-port` parameter is optional. The default port number is 61147. For more information, see the description of the `--remote-cluster-daemon` parameter later in this topic.

You can now issue the `mmnetverify` command again with the `remote-cluster` operation to run checks against the remote cluster. If you also specify the `--remote-cluster-port` parameter and the port number on which the remote-cluster daemon listens, the `remote-cluster` operation skips the first phase of the search and immediately starts the second phase. For example,

```
mmnetverify remote-cluster --remote-cluster-port <PortNumber> --cluster <clusterName>
<nodeName>
```

For more information, see the entry for the `remote-cluster` operation in Table 27 on page 554 later in this topic.

`--remote-cluster-daemon` [`--remote-cluster-port` *PortNumber*]
Causes the `mmnetverify` command to start a remote-cluster daemon on the node where the command is issued. The purpose of this daemon is to provide information about the nodes of the remote cluster to the `mmnetverify` command when it is running a `remote-cluster` operation from a node in a local cluster.

`--remote-cluster-port` *PortNumber*
Specifies the port that the remote-cluster daemon listens on. If you do not specify a port number, the daemon listens on the default port 61147.

For more information, see the description of the `--cluster` parameter earlier in this topic and the description of the `remote-cluster` operation in Table 27 on page 554 later in this topic.

The network checks

The following table lists the shortcut terms that you can specify for the network checks that are listed in Table 27 on page 554:

Note: These network checks might cause entries to the mmfs log that say that connections are being ended. These entries are expected and do not indicate problems in your system.

Table 26. Shortcut terms for network checks

Shortcut	Checks that are performed
`local`	`interface`
`connectivity`	`resolution`, `ping`, `shell`, and `copy`
`port`	`daemon-port`, `sdserv-port`, and `tsccmd-port`
`data`	`data-small`, `data-medium`, and `data-large`
`bandwidth`	`bandwidth-node` and `bandwidth-cluster`
`protocol`	`protocol-ctdb` and `protocol-object`
`flood`	`flood-node` and `flood-cluster`
`rdma`	`rdma-connectivity`
`all`	All checks except `flood-node` and `flood-cluster`

The following table lists the parameters that you can specify for network checks. Separate these parameters with a blank on the command line. For example, the following command runs the `interface` and `copy` checks on the local node against all the nodes in the cluster:

```
mmnetverify interface copy
```

Table 27. Network checks

Command-line option	Test description	Test items
`interface`	Network interface configuration	The local node's daemon and admin network addresses are enabled.
`resolution`	Host name resolution	The command verifies the following conditions: • The target node's name and daemon node name can be resolved on the local node and they resolve to the same address. • The target node's IP address resolves to either the node name or the daemon node name.
`ping`	Network connectivity via ping	The local node can ping the target node with its name, daemon node name, rel_hostname, admin_shortname, and IP address entries in the `mmsdrfs` file.
`shell`	Remote shell command	The command verifies the following conditions: • The local node can issue remote shell commands to the target node's admin interface without requiring a password. • The target node's daemon and admin names refer to the same node.
`copy`	Remote copy	The local node can issue a remote copy command to the target node's admin interface without requiring a password.
`time`	Date and time	The time and date on the local node and target node do not differ by a wide margin.
`daemon-port`[1]	GPFS daemon connectivity	The target node can establish a TCP connection to the local node on the `mmfsd` daemon port of the local node: • The target node uses the port that is specified in the cluster configuration property `tscTcpPort`. The default value is `1191`. • If `mmfsd` and `mmsdrserv` are not running on the local node, the command starts an echo server on the daemon port of the local node for this test.
`sdrserv-port`[1]	GetObject daemon connectivity	The target node can establish a TCP connection directed to the local node on the port that is specified in the cluster configuration property `mmsdrservPort`: • The default value of this port is the value that is specified in the cluster configuration property `tscTcpPort`. • If `mmfsd` and `mmsdrserv` are not running on the local node, the command starts an echo server on the daemon port of the local node for this test.

Table 27. Network checks (continued)

Command-line option	Test description	Test items
`tsccmd-port`[1]	TS-command connectivity	The target node can establish a TCP connection directed to the local node on a port in the range that is specified in the `tscCmdPortRange` property: • The command starts an echo server on the local node for this test. • If the `tscCmdPortRange` property is set, then the echo server listens on a port in the specified range. • If not, then the echo server listens on an ephemeral port that is provided by the operating system.
`data-small`[1]	Small data exchange	The target node can establish a TCP connection to the local node and exchange a series of small-sized data messages without network errors: • The command starts an echo server on the local node for this test. • If the `tscCmdPortRange` property is set, then the echo server listens on a port in the specified range. • If not, then the echo server listens on an ephemeral port that is provided by the operating system.
`data-medium`[1]	Medium data exchange	The target node can establish a TCP connection to the local node and exchange a series of medium-sized data messages without network errors: • The command starts an echo server on the local node for this test. • If the `tscCmdPortRange` property is set, then the echo server listens on a port in the specified range. • If not, then the echo server listens on an ephemeral port that is provided by the operating system.
`data-large`[1]	Large data exchange	The target node can establish a TCP connection to the local node and exchange a series of large-sized data messages without network errors: • The command starts an echo server on the local node for this test. • If the `tscCmdPortRange` property is set, then the echo server listens on a port in the specified range. • If not, then the echo server listens on an ephemeral port that is provided by the operating system.
`bandwidth-node`[1]	Network bandwidth one-to-one	The target node can establish a TCP connection to the local node and send a large amount of data with adequate bandwidth: • Bandwidth is measured on the target node. • If the `min-bandwidth` parameter was specified on the command line, the command verifies that the actual bandwidth exceeds the specified minimum bandwidth.

Table 27. Network checks (continued)

Command-line option	Test description	Test items
`bandwidth-cluster`[1]	Network bandwidth many-to-one	All target nodes can establish a TCP connection to the local node and send a large amount of data in parallel: • The bandwidth is measured on each target node. • If the `min-bandwidth` parameter was specified on the command line, the command verifies that the actual bandwidth exceeds the specified minimum bandwidth. • None of the target nodes has a significantly smaller bandwidth than the other target nodes.
`gnr-bandwidth`	Overall bandwidth	All target nodes can establish a TCP connection to the local node and send data to it. • The bandwidth is measured on each target node. • If the `min-bandwidth` parameter is specified on the command line, the command verifies that the actual bandwidth exceeds the specified minimum bandwidth. Note the following differences between this test and the `bandwidth-cluster` test: • The total bandwidth is measured rather than the bandwidth per node. • All target nodes must be active and must participate in the test. • The bandwidth value that is reported does not include ramp-up time for TCP to read full capability.
`flood-node`	Flood one-to-one	When the local node is flooded with datagrams, the target node can successfully send datagrams to the local node: • The target node tries to flood the local node with datagrams. • The command records packet loss. • The command verifies that some of the datagrams were received.
`flood-cluster`	Flood many-to-one	When the local node is flooded with datagrams from all the target nodes in parallel, each target node can successfully send datagrams to the local node: • The command records packet loss for each target node. • The command checks that each target node received some datagrams. • The command checks that none of the target nodes has a packet loss significantly higher than the other nodes.

Table 27. Network checks (continued)

Command-line option	Test description	Test items
`remote-cluster` [`--remote-cluster-port` *PortNumber*]	Remote cluster connectivity	The local node can establish a connection with the nodes in the specified remote clusters. By default this operation tests each remote cluster that is listed in the `mmsdrfs` file on the local node. You can specify more clusters to test with the `--cluster` command line parameter. This operation runs the following tests against each node in a remote cluster: • Host name resolution: – The target node's name and daemon node name can be resolved on the local node and they resolve to the same address. – The target node's IP address resolves to either the node name or the daemon node name. • Network connectivity via ping: – The local node can ping the target node with its name, daemon node name, rel_hostname, admin_shortname, and IP address entries in the `mmsdrfs` file of the target node. • GPFS daemon connectivity: – The local node can establish a TCP connection to the target node on the `mmfsd` daemon port of the target node: - The local node uses the port that is specified in the cluster configuration property `tscTcpPort`. The default value is `1191`. **Note:** Either the `mmfsd` daemon or the `mmsdrserv` daemon must be running on the remote node. (In normal operation one of these daemons is always running on a node so long as IBM Spectrum Scale is installed.) Otherwise the test fails. If you specify the `--remote-cluster-port` parameter with a port number, the remote-cluster operation skips the first phase of its search for information about a remote cluster and immediately begins the second phase of its search. For more information, see the description of the `--cluster` parameter earlier in this topic. This network check is performed only if all of the following conditions are true: • The local cluster version is IBM Spectrum Scale 5.0.2 or later. • At least one remote cluster is defined either in the `mmsdrfs` file or in a `--cluster` command-line parameter. • One of the following conditions is true: – A contact node in the remote cluster can be reached via passwordless `ssh`. – A remote-cluster daemon is running on a contact node in the remote cluster and the port that the daemon listens on is not blocked by a firewall.

Table 27. Network checks (continued)

Command-line option	Test description	Test items
`protocol-ctdb`	CTDB port connectivity	The target node can establish a connection with the local node through the CTDB port. SMB uses CTDB for internode communications. The command does not run this test in the following situations: • The SMB protocol service is not enabled. • The local node or a target node is not a CES-enabled node. A node is considered to be CES-enabled if it is in an existing cluster and if it was enabled with the `mmchnode --ces-enable` command. This requirement is not enforced in the following situations: – You override the requirement by specifying the `--ces-override` option. – You specify the `--configuration-file` parameter.
`protocol-object`	Object-protocol connectivity	The target node can establish a connection with the local node through the ports that are used by the object server, the container server, the account server, the object-server-sof, and the keystone daemons. The command does not run this test in the following situations: • The Object protocol service is not enabled. • The local node or a target node is not a CES-enabled node. A node is considered to be CES-enabled if it is in an existing cluster and if it was enabled with the `mmchnode --ces-enable` command. This requirement is not enforced in the following situations: – You override the requirement by specifying the `--ces-override` option. – You specify the `--configuration-file` parameter.
`rdma-connectivity`	RDMA connectivity	The command verifies the following conditions on nodes on which RDMA is configured: • The configuration that is specified in the `verbPorts` cluster attribute matches the active InfiniBand interfaces on the nodes. • The nodes can connect to each other through the active InfiniBand interfaces. For more information, see the `verbPorts` cluster attribute in the topic “mmchconfig command” on page 170. **Note:** These checks require that the commands `ibv_devnfo` and `ibtracert` be installed on the nodes on which RDMA is configured.

[1]For this type of test, if a list of subnets is specified, the command searches the list sequentially for a subnet that both the local node and the target node are connected to. If the command finds such a subnet, the command runs the specified test across the subnet. You can specify a list of subnets in the **subnets** attribute in the **mmchconfig** command, in the **subnets** entry of the **mmnetverify** configuration file, or in the **--subnets** parameter of the **mmnetverify** command. For more information, see the description of the **--subnets** parameter earlier in this topic.

Exit status

0
The command completed successfully and all the tests were completed successfully.

1
The command encountered problems with options or with running tests.

2
The command completed successfully, but one or more tests was unsuccessful.

Security

You must have root authority to run the `mmnetverify` command.

The node on which you enter the command must be able to execute remote shell commands on any other administration node in the cluster. It must be able to do so without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

Examples

1. The following command runs all checks, except `data-large`, `flood-node`, and `flood-cluster`, from the node where you enter the command against all the nodes in the cluster:

```
# mmnetverify
```

2. The following command runs connectivity checks from the node where you enter the command against all the nodes in the cluster:

```
# mmnetverify connectivity
```

3. The following command runs connectivity checks from nodes c49f04n11 and c49f04n12 against all the nodes in the cluster:

```
# mmnetverify connectivity -N c49f04n11,c49f04n12
```

4. The following command runs all checks, except `flood-node` and `flood-cluster`, from the node where you enter the command against nodes c49f04n11 and c49f04n12:

```
# mmnetverify all --target-nodes c49f04n11,c49f04n12
```

5. The following command runs network port checks on nodes c49f04n07 and c49f04n08, each checking against nodes c49f04n11 and c49f04n12:

```
# mmnetverify port -N c49f04n07,c49f04n08 --target-nodes c49f04n11,c49f04n12
```

See also

- "mmlsqos command" on page 530

Location

`/usr/lpp/mmfs/bin`

mmnfs command

Manages NFS exports and configuration.

Synopsis

```
mmnfs config list  [--exportdefs][-Y]
```

or

```
mmnfs config change Option=Value:Option=Value1,Value2:Option=Value...
                              [--force]
```

or

```
mmnfs export list [-Y] [{--nfsdefs Path}|{--nfsdefs-match Path}]
```

or

```
mmnfs export load ExportCFGFile
```

or

```
mmnfs export add Path  [--client "ClientOption[;ClientOption...]"]
                         [--pseudo Path][--dry-run]
```

or

```
mmnfs export remove Path [--force]
```

or

```
mmnfs export change Path {--nfsadd "ClientOption[;ClientOption...]"|
                          --nfsremove Client[,Client...]|
                          --nfschange "ClientOption[;ClientOption...]"}
                          [--nfsposition {PositionIndex | Client}]
                          [--dry-run]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmnfs export` commands to add, change, list, load, or remove NFS export declarations for IP addresses on nodes that are configured as CES types.

Use the `mmnfs config` commands to list and change NFS configuration.

The protocol functions provided in this command, or any similar command, are generally referred to as CES (Cluster Export Services). For example, protocol node and CES node are functionally equivalent terms.

Parameters

export
Manages the NFS export configuration for the cluster with one of the following actions:

add

Creates a new configuration file for the NFS server in case it does not yet exist. If there is already an export configuration file, then it is extended with the provided additional export parameters. This export configuration file is used by the NFS server to create an NFS export for the *Path* so that clients can connect to it. If there is already an existing export for the *Path* then an error is shown. Each export configuration set has internally its own unique identifier number. This number is automatically incremented for each added export. The `mmnfs export add` command attempts to add the new export also to running NFS server instances, and can fail if one or more instances are not running. This is not a critical issue, because the configuration changes are made in the repository and are applied later when restarting the NFS server instance.

The authentication method must be established before an NFS export can be defined.

The export *Path* must be an existing path in the GPFS file system.

Note: The paths that are not within the GPFS file system cannot be exported using the commands.

Creating nested exports (such as `/path/to/folder` and `/path/to/folder/subfolder`) is not recommended because doing so might lead to serious issues in data consistency. Remove the higher-level export that prevents the NFSv4 client from descending through the NFSv4 virtual file system path. In case nested exports cannot be avoided, ensure that the export with the common path, called as the top-level export, has all the permissions for this NFSv4 client. Also NFSv4 client that mounts the parent (`/path/to/folder`) export does not see the child export subtree (`/path/to/folder/inside/subfolder`) unless the same client is explicitly allowed to access the child export as well.

--client {*ClientOption*[;*ClientOption*...]}

Declares the client-specific settings. You can specify a list of one or more client definitions. To avoid incorrect parsing by the interpreter, put quotation marks around the argument list. For a list of client definitions that can be specified with the `--client` option, see List of supported client options for the mmnfs export {add | change} command.

Therefore, some export configuration commands might allow multiple client declarations, and they have separators to distinguish them.

The following separators can be used:

- **Colon** to separate multiple allowed values for a specified attribute. For example, the key/value pair "Protocols=3:4" allows the NFS protocols v3 and v4 to be declared for an export.
- **Comma** to separate key/value pairs within a client declaration list and a **Semicolon** to separate client declaration lists.

  For example:

```
--client "192.0.2.0/20(Access_Type=RW,Protocols=3:4); \
198.51.100.0/20(Access_Type=RO,Protocols=3:4,Transports=TCP:UDP)"
```

Note: NFS service restarts after the first export creation.

Note: To take advantage of the GPFS independent fileset features such as quotas, snapshots, and data management, the export paths can be made from GPFS filesets (either dependent or independent).

--pseudo

Specifies the path name the NFSv4 client uses to locate the directory in the server's file system tree.

Note: This option is applicable to NFSv4 mounts only.

remove

Removes the requested export from the configuration file and also from running NFS server instances, and might fail if one or more instances are not running.

Note: This command does not remove data.

The `--force` option is not used.

change
Modifies an existing export configuration for the export that is specified by *Path*, if the export exists. If the export does not exist, then an error is shown.

Note: Only client-related attributes are modified by this command, but not the basic export settings such as path.

The **--nfsposition** flag can be only used together with either `--nfsadd` or `--nfschange`. It cannot be used stand-alone or together with `--nfsremove`. When `--nfsadd` and `--nfsremove` are given on the same command line, then the remove procedure is run first internally.

`--nfsadd` {*ClientOption*[;*ClientOption*...]}
Adds a new client declaration for the specified *Path*. You can list one or more client definitions. To avoid incorrect parsing by the interpreter, put quotation marks around the argument list. For a list of client definitions that can be specified with the `--client` option, see List of supported client options for the mmnfs export {add | change} command.

`--nfsremove` {*Client*[,*Client*...]}
Removes the NFS client that is specified by *Client* from the export configuration for the *Path*. *Client* can be a single client specifier, or a comma-separated list of client specifiers. If a client is the only one within a client declaration section in the configuration file, then this client section is removed.

Note: When the last client within a client declaration is removed, then the export section is also removed.

`--nfschange` {*ClientOption*[;*ClientOption*...]}
Modifies an existing client declaration for the specified *Path*. You can specify a list of one or more client definitions. To avoid incorrect parsing by the interpreter, put quotation marks around the argument list. For a list of client definitions that can be specified with the `--client` option, see List of supported client options for the mmnfs export {add | change} command. If a client specifier declared in *ClientOption* matches a client declaration section, which has further NFS clients that are assigned, then this section is split into a section containing that client specifier together with the modified attributes, and a section with the remaining client specifiers and their original attributes.

The `--nfsposition` option can be used to locate this modified entry.

`--nfsposition` {*PositionIndex* | *Client*}
This option can be used only together with `--nfsadd` or `--nfschange`. It reorders the client declaration section within the export declaration file. The sequence of client entries can be important when there are different client sections affecting the same NFS client. For example, NFS position can be important when defining export attributes for a specific NFS client, and '*' for all NFS clients, given the same export path.

PositionIndex
This must be an unsigned numeric value, and it indicates the absolute sequence number after reordering the client declarations. The indexing is zero-based. 0 means the top of the list, 1 means the second entry, and so on. Any number higher than the number of existing entries indicates the end-of-list position.

Client
The reference client specifier to indicate the new position of the current client entry. The modified client entry is placed at the location where the *Client* reference declaration is, and that section is ordered below next to the modified client section.

list
Lists the declared exports and clients. The sequence of rows in the output for a given path reflects also the sequence of the internal client declaration list for each exported path. The sequence of client declarations within an export can be reordered using the `mmnfs export change` *Path* with the `--nfschange` and `--nfsposition` options. The output can be formatted human readable (default) or machine readable.

--nfsdefs *Path*
Lists the export configuration details for the specified *Path*.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

--nfsdefs-match *Path*
Lists the export configuration details of all export paths, which contain the specified string.

load
Overwrites (deletes) all existing NFS export declarations in the repository, if any. The export declarations are fetched from a file provided to the load operation, which could contain a larger number of export declarations. Some basic format checks are done during export load. After loading export declarations from a file, the NFS service is restarted across all the nodes in the cluster.

ExportCFGFile
The file name for the new exports declarations. This file is loaded and stored in the repository to be published on all CES nodes running the NFS server. This load procedure can be used to load a set of export declarations and that removes any previous configuration. The NFS servers are restarted in order to apply the changes.

List of supported client options for the `mmnfs export {add | change}` command:

ACCESS_TYPE
Allowed values are `none`, `RW`, `RO`, `MDONLY`, and `MDONLY_RO`. The default value is `none`.

PROTOCOLS
Allowed values are `3`, `4`, `NFS3`, `NFS4`, `V3`, `V4`, `NFSv3`, and `NFSv4`. The default value is `3,4`.

TRANSPORTS
Allowed values are `TCP` and `UDP`. The default value is `TCP`.

ANONYMOUS_UID
Allowed values are between `-2147483648` and `4294967295`. The default value is `-2`.

ANONYMOUS_GID
Allowed values are between `-2147483648` and `4294967295`. The default value is `-2`.

SECTYPE
Allowed values are `none`, `sys`, `krb5`, `krb5i`, and `krb5p`. The default value is `sys`.

PRIVILEGEDPORT
Allowed values are `true` and `false`. The default value is `false`.

MANAGE_GIDS=true
In this configuration, NFS server discards the list of GIDs passed via the RPC and fetch the group information. For getting the group list, the server uses the default authentication configured on the system (CES node). So, depending on the configuration, the list is obtained by using (/etc/passwd + /etc/groups) or from LDAP or from AD. To reach AD and get the list of GIDs, the client must use sec=krb5, else NFS server is not able to get the list from AD.

MANAGE_GIDS=false
NFS server uses list of GIDs passed by RPC.

SQUASH
Allowed values are `root`, `root_squash`, `all`, `all_squash`, `allsquash`, `no_root_squash`, `none`, and `noidsquash`. The default value is `root_squash`.

NFS_COMMIT
Allowed values are `true` and `false`. The default value is `false`.

Important: Use `NFS_COMMIT` very carefully because it changes the behavior of how transmitted data is committed on the server side to NFS v2 like sync-mode on every write action.

CLIENTS
Allowed values are IP addresses in IPv4 or IPv6 notations, hostnames, netgroups, or * for all. A netgroup name must not start with a numeric character and otherwise must only contain underscore and/or hyphens ("[a-zA-Z_][0-9a-zA-Z_\- .]*"). The default value is `*`.

config
Manages NFS configuration for a CES cluster:

list
Displays the NFS configuration parameters and their values. This command also displays all the default export configurations. This is used as the defaults by the `mmnfs export add` command if no other client attributes are specified. The output can be formatted to be human readable or machine readable.

--exportdefs
If this option is specified, the command displays the default export configuration parameters.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

change
Modifies the NFS configuration parameters. NFS is restarted across all the nodes on which NFS is running, when this command is executed. Only some configuration options can be modified by this command.

The configuration options that can be modified and their allowed values are as follows:

NFS_PROTOCOLS
Allowed values are 3, 4, NFS3, NFS4, V3, V4, NFSv3, and NFSv4. The default value is `3,4`.

MINOR_VERSIONS
Allowed values are 0 and 1. The default value is 0 that implies NFS 4.0 is supported. The value 1 enables the NFS 4.1 support.

MNT_PORT
Specifies the port for the NFSv3 Mount protocol. Allowed values are between `0` and `65535`. The default value is `0`.

NLM_PORT
Specifies the NLM port for NFSv3. Allowed values are between `0` and `65535`. The default value is `0`.

RQUOTA_PORT
Specifies the RQUOTA port for NFSv3. Allowed values are between `0` and `65535`. The default value is `0`.

STATD_PORT
Specifies the STATD port for NFSv3. Allowed values are between `0` and `65535`. The default value is `0`.

LEASE_LIFETIME
Specifies the value in seconds during which an NFS server requires an NFS client to do some operation to renew its lease. This is specific to NFSV4. Allowed values are between `0` and `120`. The default value is `60`.

GRACE_PERIOD
Specifies the time in seconds during which an NFS client is required to reclaim its locks, in the case of NFSv3, and additionally its state, in the case of NFSv4. Allowed values are between `0` and `180`. The default value is `90`.

DOMAINNAME
String. The default value is the `"host's fully-qualified DNS domain name"`.

IDMAPD_DOMAIN
String. Domain in the ID Mapd configuration. The default value is the host's fully-qualified DNS domain name.

LOCAL_REALMS
String. Local-Realms in the ID Mapd configuration. The default value is the host's default realm name.

LOG_LEVEL
Allowed values are NULL, FATAL, MAJ, CRIT, WARN, EVENT, INFO, DEBUG, MID_DEBUG, and FULL_DEBUG. The default value is EVENT.

ENTRIES_HWMARK
The high water mark for NFS cache entries. Beyond this point, NFS tries to evict some objects from its cache. The default is 1500000.

RPC_IOQ_THRDMAX
Specifies the maximum number of RPC threads that process NFS requests from NFS clients. The valid values are 2 - 131072. The default value is 512.

If the system has high bandwidth but high latency, you can increase the value of this parameter to improve the system performance. However, increasing this parameter causes Ganesha to use more memory under a heavy workload.

The **NB_WORKER** parameter of Ganesha 2.3 and 2.5 is obsolete, use the **RPC_IOQ_THRDMAX** parameter in Ganesha 2.7.

Note: Specifying a port number in the NFS service configuration with the value of '0' means that the service is picking a port number dynamically. This port number might change across service restarts. If a firewall is to be established between the NFS server and the NFS clients, specific port number can be configured via the command to establish discrete firewall rules. Note that the NFS_PORT 2049 is a well-known and established convention as NFS servers and clients typically expect this port number. Changing the NFS service port numbers impacts existing clients and a remount of the client is required.

The export defaults that can be set are:

ACCESS_TYPE
Allowed values are none, RW, RO, MDONLY, and MDONLY_RO. The default value is none.

Note: Changing this option to any value other than none exposes data to all NFS clients that can access your network, even if the export is created using `add --client` *ClientOptions* to limit that clients access. All clients, even if not declared with the `--client` in the `mmnfs export add`, has access to the data. The global value applies to an unseen *, even if `showmount -e` *CESIP* does not display it. Use caution if you change it in this global definition.

ANONYMOUS_UID
Allowed values are between -2147483648 and 4294967295. The default value is -2.

ANONYMOUS_GID
Allowed values are between -2147483648 and 4294967295. The default value is -2.

MANAGE_GIDS
Allowed values are true and false. The default value is false.

NFS_COMMIT
Allowed values are true and false. The default value is false.

Important: Use NFS_COMMIT very carefully because it changes the behavior of how transmitted data is committed on the server side to NFS v2 like sync-mode on every write action.

PRIVILEGEDPORT
Allowed values are true and false. The default value is false.

PROTOCOLS
Allowed values are 3, 4, NFS3, NFS4, V3, V4, NFSv3, and NFSv4. The default value is 3,4.

SECTYPE
Allowed values are none, sys, krb5, krb5i, and krb5p. The default value is sys.

SQUASH
Allowed values are root, root_squash, all, all_squash, allsquash, no_root_squash, none, and noidsquash. The default value is root_squash.

Note: Changing this option to no_root_squash exposes data to root on all NFS clients, even if the export is created using add --client *ClientOptions* to limit the root access of that client.

TRANSPORTS
Allowed values are TCP and UDP. The default value is TCP.

If export defaults are set, then new exports that are created pick up the export default values. If an export configuration value is not specified, the current export default value is used.

Note: For exports created before IBM Spectrum Scale 5.0.2, the export default value at the time of its creation is used.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the mmnfs command.

The node on which the command is issued must be able to execute remote shell commands on any other CES node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To create an NFS export (using a netgroup), issue this command:

```
# mmnfs export add /mnt/gpfs0/netgrouppath \
--client "@netgroup(Access_Type=RO,Squash=allsquash)"
```

The system displays output similar to this:

```
The NFS export was created successfully.
```

Note: To specify an NFS client, use an IP address in IPv4 or IPv6 notation, hostname, netgroup, or * for all.

2. To create an NFS export (using a client IP), issue this command:

```
# mmnfs export add /mnt/gpfs0/netgrouppath --client "192.0.2.0/20(Access_Type=RW)"
```

The system displays output similar to this:

```
The NFS export was created successfully.
```

3. To add a client definition, issue these commands:

```
# mmnfs export list --nfsdefs /gpfs/fs1/export_1
```

The system displays output similar to this:

```
Path            Delegations  Clients  Access_Type Protocols Transports Squash  Anonymous_uid Anonymous_gid SecType PrivilegedPort DefaultDelegations Manage_Gids
```

```
NFS_Commit
------------------------------------------------------------------------------------------------------------------------------------------------------
-----
/gpfs/fs1/export_1   none      192.0.2.8    RW       3,4      TCP       ROOT_SQUASH   -2          -2       SYS     FALSE         none            FALSE
FALSE
```

Now add a client definition that is more restrictive to a different client, 198.51.100.10, by issuing the command:

```
# mmnfs export change /gpfs/fs1/export_1 \
--nfsadd "198.51.100.10(Access_Type=MDONLY,Squash=allsquash)"
```

```
# mmnfs export list
```

The system displays output similar to this:

```
Path                          Delegations    Clients
-------------------------------------------------
/gpfs/fs1/controller          none           *
/gpfs/fs1/export_1            none           192.0.2.8
/gpfs/fs1/export_1            none           198.51.100.10
```

Now add a client definition that is very permissive but we want it to be last on the list so that the more restrictive attributes for client 192.0.2.8 take precedence for that one client, by issuing the command:

```
# mmnfs export change /gpfs/fs1/export_1 \
--nfsadd "192.0.2.0/20(Access_Type=RW,Squash=no_root_squash)" --nfsposition 4
```

```
# mmnfs export list --nfsdefs /gpfs/fs1/export_1
```

The system displays output similar to this:

```
Path              Delegations Clients    Access Protocols Transports Squash         Anonymous Anonymous SecType Privileged  Default
Manage_Gids NFS_Commit
                                           _Type                                      _uid      _gid              Port
Delegation
------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------
/gpfs/fs1/export_1   none      192.0.2.8      RW       3,4       TCP     ROOT_SQUASH      -2       -2        SYS     FALSE    none
FALSE       FALSE
/gpfs/fs1/export_1   none      198.51.100.10  MDONLY   3,4       TCP     allsquash        -2       -2        SYS     FALSE    none
FALSE       FALSE
/gpfs/fs1/export_1   none      192.0.2.0/20   RW       3,4       TCP     no_root_squash   -2       -2        SYS     FALSE    none
FALSE       FALSE
```

4. To remove an NFS export, issue this command:

   ```
   # mmnfs export change /mnt/gpfs0/somepath  --nfsremove "1.2.3.1"
   ```

   The system displays output similar to this:

   ```
   The NFS export was changed successfully.
   ```

   Note: This command removes a single client definition, for the IP "1.2.3.1", from the NFS export. If this is the last client definition for the export, then the export is also removed.

5. To modify an NFS export, issue this command:

   ```
   # mmnfs export change  /mnt/gpfs0/p1 \
   --nfschange "203.0.113.2(Access_Type=RO)" --nfsposition 0
   ```

   The system displays output similar to this:

   ```
   The NFS export was changed successfully.
   ```

6. To list the NFS configuration, issue this command:

   ```
   # mmnfs config list
   ```

   The system displays output similar to this:

```
   NFS Ganesha Configuration:
   ==========================
   NFS_PROTOCOLS: 3,4
   NFS_PORT: 2049
   MNT_PORT: 0
   NLM_PORT: 0
   RQUOTA_PORT: 0

   LEASE_LIFETIME: 60
   GRACE_PERIOD: 90
   DOMAINNAME: VIRTUAL1.COM
   DELEGATIONS: Disabled
   ==========================

   STATD Configuration
   ==========================
   STATD_PORT: 0
   ==========================

   CacheInode Configuration
   ==========================
   ENTRIES_HWMARK: 1500000
   ==========================

   Export Defaults
   ==========================
   ACCESS_TYPE: NONE
   PROTOCOLS: 3,4
   TRANSPORTS: TCP
   ANONYMOUS_UID: -2
   ANONYMOUS_GID: -2
   SECTYPE: SYS
   PRIVILEGEDPORT: FALSE
   MANAGE_GIDS: FALSE
   SQUASH: ROOT_SQUASH
   NFS_COMMIT: FALSE
   ==========================

   Log Configuration
   ==========================
   LOG_LEVEL: EVENT
   ==========================

   Idmapd Configuration
   ==========================
   LOCAL-REALMS: LOCALREALM
   DOMAIN: LOCALDOMAIN
   ==========================
```

7. To change STATD_PORT configuration, issue this command (When a port is assigned, STATD is started on the given port):

```
# mmnfs config change STATD_PORT=32765
```

The system displays output similar to this:

```

NFS Configuration successfully changed. NFS server restarted on all NFS nodes
on which NFS server is running.
```

8. To list NFS exports, issue this command:

```
# mmnfs export list
```

The system displays output similar to this:

```

Path                Delegations    Clients
-----------------------------------------
/gpfs/fs1/controller none           *
/gpfs/fs1/export_1   none           *
/gpfs/fs1/export_2   none           *
```

9. To list NFS exports pertaining to */mnt/gpfs0/p1*, issue this command:

```
# mmnfs export list --nfsdefs /mnt/gpfs0/p1
```

The system displays output similar to this:

```
Path           Delegations Clients     Access_ Protocols Transports Squash      Anonymous Anonymous  SecType  Privileged Default   MANAGE
NFS
                                       Type                                      _uid      _gid                Port    Delegations  _Gids
_Commit
-------------------------------------------------------------------------------------------------------------------------------------------
-------
/mnt/gpfs0/p1  none        203.0.113.2 RO      3,4       TCP        ROOT_SQUASH  -2        -2         SYS      FALSE      none     FALSE
FALSE
/mnt/gpfs0/p1  none        *           RW      3,4       TCP        ROOT_SQUASH  -2        -2         SYS      FALSE      none     FALSE
FALSE
/mnt/gpfs0/p1  none        203.0.113.1 RO      3,4       TCP        ROOT_SQUASH  -2        -2         SYS      FALSE      none     FALSE
FALSE
```

10. To list export configuration details for the path */gpfs/gpfs1/export1*, issue this command:

```
# mmnfs export list --nfsdefs /gpfs/gpfs1/export1
```

The system displays output similar to this:

```
Path               Deleg.      Clients   Access_Type Protocols Transports Squash         Anon_uid Anon_gid SecType PrivPort DefDeleg
Manage_Gids NFS_Commit
-------------------------------------------------------------------------------------------------------------------------------------------
---------------
/gpfs/gpfs1/export1 NONE        c49f04n10 RO          3,4       TCP        ROOT_SQUASH    -2       -2       SYS     FALSE    NONE
FALSE       FALSE
/gpfs/gpfs1/export1 NONE        *         RW          3         TCP        ROOT_SQUASH    -2       -2       SYS     FALSE    NONE
FALSE       FALSE
/gpfs/gpfs1/export1 NONE        c49f04n12 RO          3,4       TCP        ROOT_SQUASH    -2       -2       SYS     FALSE    NONE
FALSE       FALSE
/gpfs/gpfs1/export1 NONE        c49f04n11 NONE        3,4       TCP        NO_ROOT_SQUASH -2       -2       SYS     FALSE    NONE
FALSE       FALSE
```

11. To list export configuration details for all paths that contain the string - */gpfs/gpfs1/export1*, issue this command:

```
# mmnfs export list --nfsdefs-match /gpfs/gpfs1/export1
```

The system displays output similar to this:

```
Path               Deleg.      Clients   Access_Type Protocols Transports Squash         Anon_uid Anon_gid SecType PrivPort DefDeleg
Manage_Gids NFS_Commit
-------------------------------------------------------------------------------------------------------------------------------------------
---------------
/gpfs/gpfs1/export1 NONE        c49f04n10 RO          3,4       TCP        ROOT_SQUASH    -2       -2       SYS     FALSE    NONE
FALSE       FALSE
/gpfs/gpfs1/export1 NONE        *         RW          3         TCP        ROOT_SQUASH    -2       -2       SYS     FALSE    NONE
FALSE       FALSE
/gpfs/gpfs1/export1 NONE        c49f04n12 RO          3,4       TCP        ROOT_SQUASH    -2       -2       SYS     FALSE    NONE
FALSE       FALSE
/gpfs/gpfs1/export1 NONE        c49f04n11 NONE        3,4       TCP        NO_ROOT_SQUASH -2       -2       SYS     FALSE    NONE
FALSE       FALSE
/gpfs/gpfs1/export11NONE        *         NONE        3,4       TCP        ROOT_SQUASH    -2       -2       SYS     FALSE    NONE
FALSE       FALSE
```

12. To list exports of a specific fileset, issue this command:

```
# mmnfs export list -n /gpfs/FS1/fset_io
```

The system displays output similar to this:

```
Path  Delegations Clients     Access_Type Protocols Transports Squash Anonymous_uid Anonymous_gid SecType PrivilegedPort DefaultDelegations
Manage_Gids NFS_Commit
-------------------------------------------------------------------------------------------------------------------------------------------
---------------------
/gpfs/FS1/fset_io NONE        @host.linux       RW          3,4       TCP        NO_ROOT_SQUASH -2   -2   SYS     FALSE
NONE        FALSE       FALSE
/gpfs/FS1/fset_io NONE        @host.hsn61_linux RW          3,4       TCP        NO_ROOT_SQUASH -2   -2   SYS     FALSE
NONE        FALSE       FALSE
```

13. To enable NFS 4.1 version, issue this command:

```
# mmnfs config change MINOR_VERSIONS=0,1
```

See also

- “mmces command” on page 133
- “mmchconfig command” on page 170
- “mmlscluster command” on page 492
- “mmlsconfig command” on page 495
- “mmobj command” on page 573
- “mmsmb command” on page 709
- “mmuserauth command” on page 738

Location

`/usr/lpp/mmfs/bin`

mmnsddiscover command

Rediscovers paths to the specified network shared disks.

Synopsis

```
mmnsddiscover [-a | -d "Disk[;Disk...]" | -F DiskFile] [-C ClusterName]
   [-N {Node[,Node...] | NodeFile | NodeClass}]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmnsddiscover` command is used to rediscover paths for GPFS NSDs on one or more nodes. If you do not specify a node, GPFS rediscovers NSD paths on the node from which you issued the command.

On server nodes, `mmnsddiscover` causes GPFS to rediscover access to disks, thus restoring paths which may have been broken at an earlier time. On client nodes, `mmnsddiscover` causes GPFS to refresh its choice of which NSD server to use when an I/O operation occurs.

In general, after the path to a disk is fixed, the `mmnsddiscover` command must be first run on the server that lost the path to the NSD. After that, run the command on all client nodes that need to access the NSD on that server. You can achieve the same effect with a single `mmnsddiscover` invocation if you utilize the **-N** option to specify a node list that contains all the NSD servers and clients that need to rediscover paths.

Parameters

-a
Rediscovers paths for all NSDs. This is the default.

-d "*DiskName*[;*DiskName*]"
Specifies a list of NSDs whose paths are to be rediscovered.

-F *DiskFile*
Specifies a file that contains the names of the NSDs whose paths are to be rediscovered.

-C *ClusterName*
Specifies the name of the cluster to which the NSDs belong. This defaults to the local cluster if not specified.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies the nodes on which the rediscovery is to be done.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmnsddiscover` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To rediscover the paths for all of the NSDs in the local cluster on the local node, issue the command:

```
mmnsddiscover
```

   The system displays output similar to:

```
mmnsddiscover:  Attempting to rediscover the disks.
   This may take a while ...

mmnsddiscover:  Finished.
```

2. To rediscover the paths for all of the NSDs in the local cluster on all nodes in the local cluster, issue the command:

```
mmnsddiscover -a -N all
```

   The system displays output similar to:

```
mmnsddiscover:  Attempting to rediscover the disks.
   This may take a while ...

mmnsddiscover:  Finished.
```

3. To rediscover the paths for a given list of the NSDs on a node in the local cluster, issue the command:

```
mmnsddiscover -d "gpfs1nsd;gpfs2nsd" -N c6f2c2vp5
```

   The system displays output similar to:

```
mmnsddiscover: Attempting to rediscover the disks. This may take a while ...
c6f2c2vp5.ppd.pok.ibm.com: GPFS: 6027-1805 [N] Rediscovered nsd server access to gpfs1nsd.
c6f2c2vp5.ppd.pok.ibm.com: GPFS: 6027-1805 [N] Rediscovered nsd server access to gpfs2nsd.
mmnsddiscover: Finished.
```

See also

- “mmchnsd command” on page 254
- “mmcrnsd command” on page 335
- “mmdelnsd command” on page 381
- “mmlsnsd command” on page 522

Location

/usr/lpp/mmfs/bin

mmobj command

Manages configuration of Object protocol service, and administers storage policies for object storage, unified file and object access, and multi-region object deployment.

Synopsis

```
mmobj swift base -g GPFSMountPoint --cluster-hostname CESHostName
                    [-o ObjFileset] [-i MaxNumInodes] [--ces-group CESGroup]
                    {{--local-keystone} | [--remote-keystone-url URL]
                    [--configure-remote-keystone]}
                    [--admin-user AdminUser] [--swift-user SwiftUser ][--pwd-file PasswordFile]
                    [--enable-file-access] [--enable-s3]  [--enable-multi-region]
                    [--join-region-file RegionFile] [--region-number RegionNumber]
```

or

```
mmobj config list --ccrfile CCRFile [--section Section [--property PropertyName]] [-Y]
```

or

```
mmobj config change --ccrfile CCRFile --section Section --property PropertyName --value Value
```

or

```
mmobj config change --ccrfile CCRFile --merge-file MergeFile
```

or

```
mmobj config  manage --version-sync
```

or

```
mmobj policy list [{--policy-name PolicyName | --policy-function PolicyFunction}] [--verbose]
[-Y]
```

or

```
mmobj policy create PolicyName [-f FilesetName]
[--file-system FilesystemName] [-i MaxNumInodes]
{[--enable-compression --compression-schedule CompressionSchedule]}
{[--enable-encryption --encryption-keyfile EncryptionKeyFileName [--force-rule-append]]}
[--enable-file-access]
```

or

```
mmobj policy change PolicyName --default
```

or

```
mmobj policy change PolicyName --deprecate State
```

or

```
mmobj policy change --add-local-region
```

or

```
mmobj policy change --remove-region-number RegionNumber
```

or

```
mmobj policy change --compression-schedule CompressionSchedule
```

or

```
mmobj file-access enable
```

or

```
mmobj file-access disable [--objectizer]
```

or

```
mmobj file-access objectize
{ --object-path ObjectPath
| --storage-policy PolicyName
[ --account-name AccountName
[ --container-name ContainerName
[ --object-name ObjectName  ]]]}
[ -N | --node NodeName ]
```

or

```
mmobj file-access link-fileset
--sourcefset-path FilesetPath
--account-name AccountName
--container-name ContainerName
--fileaccess-policy-name PolicyName
[--update-listing]
```

or

```
mmobj multiregion list  [-Y]
```

or

```
mmobj multiregion enable
```

or

```
mmobj multiregion export --region-file RegionFile
```

or

```
mmobj multiregion import --region-file RegionFile
```

or

```
mmobj multiregion remove --region-number RegionNumber --force
```

or

```
mmobj s3 enable
```

or

```
mmobj s3 disable
```

or

```
mmobj s3 list [-Y]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the **mmobj** command to modify and change the Object protocol service configuration, and to administer storage policies for Object Storage, unified file and Object access, and multi-region Object deployment.

Note: The `mmobj config list` and the `mmobj config change` commands are used to list and change the configuration values for the underlying Swift service that is stored in the Cluster Configuration Repository (CCR).

At least one CES IP address is required in the node that is running the `mmobj swift base` command to set `object_singleton_node` and `object_database_node` attributes.

- The node with the `object_database_node` attribute runs the keystone database.
- The node with the `object_singleton_node` attribute runs unique object services across the CES cluster.

You can verify the address by using the `mmces address list`. The IP address can be added by using the `mmces address add` command.

If there is an existing Object authentication that is configured, use the **mmuserauth service remove --data-access-method object** command to remove the Object authentication. Then, use the **mmces service disable OBJ** command for necessary cleanup before you run the **mmobj swift base** command again.

Object authentication can be either local or remote. If the Object authentication is local, the Keystone identity service and the Keystone database run in the CES cluster and are handled by the CES cluster. If the Object authentication is remote, the Keystone server must be fully configured and running before Object services are installed.

In the unified file and Object access environments, the `ibmobjectizer` service makes files from the file interfaces such as POSIX, NFS, and CIFS accessible through Object interfaces such as curl and SWIFT. The `ibmobjectizer` service runs periodically and makes files available for the Object interface. The **file-access** parameter of the **mmobj** command can be used to enable and disable the file-access capability and the `ibmobjectizer` service. The `ibmobjectizer` service runs on the CES IP address with the `object_singleton_node` attribute.

In a data lake environment, you can access files that are already stored in an existing fileset through Object interface through swift or curl by using the `mmobj file-access link-fileset` command. This command can be used to make the existing fileset data accessible under a unified file and Object access storage policy container. By default, this command allows the existing fileset to have Object access without updating the container and metadata. However, you can use the `--update-listing` option to update container listing with files that are existing in the linked fileset. If the `--update-listing` option of the **mmobj file-access link-fileset** command is used, then the `sourcefset-path` must be the exact fileset junction path and the fileset must be derived from the Object file system.

Note: The **mmobj file-access link-fileset** command does not enable and disable the unified file and Object access feature. It is only used to link the existing filesets to provide Object access. Unlinking the Object access-enabled filesets is not supported.

Parameters

swift
Configures the underlying Swift services.

Note: The object protocol is not supported in IBM Spectrum Scale 5.1.0.0. If you want to deploy object, install the IBM Spectrum Scale 5.1.0.1 or a later release.

base
Specifies the configuration of the Object protocol service.

-g *GPFSMountPoint*
Specifies the mount path of the GPFS file system that is used by Swift.

GPFSMountPoint is the mount path of the GPFS file system that is used by the Object store.

--cluster-hostname *CESHostName*
Specifies the host name that is used to return one of the CES IP addresses. The returned CES IP address is used in the endpoint for the identity and Object-store values that are stored in Keystone.

CESHostName is the value for cluster host name that is used in the identity and Object-store endpoint definitions in Keystone. Ideally, the host name is a host name that returns one the CES IP addresses, such as a round-robin DNS. It might also be a fixed IP of a load balancer that distributes requests to one of the CES nodes. It is not recommended to use an ordinary CES IP since all identity and Object-store requests would be routed to the single node with that address and might cause performance issues.

--local-keystone
Specifies that a new Keystone server is installed and configured locally in the cluster.

--admin-user *User*
Specifies the name of the admin user in Swift. The default value is `admin`.

--swift-user *User*
Specifies the user for the Swift services. The default value is `swift`.

--pwd-file *PasswordFile*
Specifies the file that contains the administrative user passwords that are used for Object access protocol authentication configuration. You must save the password file under `/var/mmfs/ssl/keyServ/tmp` on the node from which you are running the command. The password file is a security-sensitive file that must have the following characteristics:

- The password file must be a regular file.
- The password file must be owned by the root user.
- Only the root user must have permission to read or write it.

Important: Passwords cannot contain the following characters:

- Forward slash (/)
- Colon (:)
- Backward slash (\)
- At symbol (@)
- Dollar sign ($)
- Left curly bracket ({)
- Right curly bracket (})
- Space

The password file for Object protocol configuration must have the following format:

```
%objectauth:
     ksAdminPwd=ksAdminPwdpassword
     ksSwiftPwd=ksSwiftPwdpassword
     ksDatabasePwd=ksDatabasePassword
```

In the password file example:

objectauth
Indicates the stanza name for the object protocol.

ksAdminPwd
Specifies the Keystone administrator's password.

ksSwiftPwd
Specifies the Swift user's password. If not specified, this value defaults to the Keystone administrator's password.

ksDatabasePwd
Specifies the password for the Keystone user in the `postgres` database. If not specified, this value defaults to the Keystone administrator's password.

--remote-keystone-url *URL*
Specifies the URL to an existing Keystone service.

--configure-remote-keystone
Specifies that when you use a remote Keystone and configure the remote Keystone as necessary. The required users, roles, and endpoints that are needed by the Swift services are added to the Keystone server. Keystone authentication information needs to be specified with the `--pwd-file` flag to enable the configuration. If this flag is not specified, the remote Keystone is not modified and the administrator must add the appropriate entries for the Swift configuration after the installation is complete.

-o *ObjFileset*
Specifies the name of the fileset to be created in GPFS for the Object Storage.

ObjFileset is the name of the independent fileset that is created for the Object store. By default, `object_fileset` is created.

-i
Specifies the maximum number of inodes for the Object fileset.

MaxNumInodes
The maximum number of inodes for the Object fileset. By default, `8000000` is set.

--ces-group
Specifies the CES group that contains the IP addresses to be used for the Swift ring files. This means that you can specify a subset of the overall collection of CES IP addresses to be used by the object protocol.

--enable-s3
Sets the S3 capability (Amazon S3 API support) to `true`. By default, S3 API is not enabled.

--enable-file-access
Sets the file access capability initially to `true`. Further configuration is still necessary by using the **`mmobj file-access`** command. By default, the **`file-access`** capability is not enabled.

--enable-multi-region
Sets the multi-region capability initially to `true`. By default, multi-region capability is not enabled.

--join-region-file *RegionFile*
Specifies that this object installation joins an existing object multi-region Swift cluster. *RegionFile* is the region data file created by the **`mmobj multiregion export`** command from the existing multi-region cluster.

Note: The use of the `--configure-remote-keystone` flag is recommended so that the region-specific endpoints for this region are automatically created in Keystone.

--region-number *RegionNumber*
Specifies the Swift region number for this cluster. If it is not specified, the default value is set to 1. In a multi-region configuration, this flag is required and must be a unique region number that is not used by another region in the multi-region environment.

config
Administers the Object configuration:

list
Lists configuration values of the underlying Swift or Keystone service that is stored in the CCR.

--section *Section*
Retrieves values for the specified section only.

The section is the heading that is enclosed in brackets ([]) in the associated configuration file.

--property *PropertyName*
Retrieves values for the specified property only.

Note: Use the **-Y** parameter to display the command output in a parseable format with a colon (:) as a field delimiter.

change
Enables modifying Swift or Keystone configuration files. After you modify the configuration files, the CES monitoring framework downloads them from the CCR and distributes them to all the CES nodes in the cluster. The framework also automatically restarts the services that depend on the modified configuration files.

Note: It is recommended to not directly modify the configuration files in `/etc/swift` and `/etc/keystone` folders as they can be overwritten at any time by the files that are stored in the CCR.

--section *Section*
Specifies the section in the file that contains the parameter.

The section is the heading that is enclosed in brackets ([]) in the associated configuration file.

--property *PropertyName*
Specifies the name of the property to be set.

--value *NewValue*
Specifies the value of the *PropertyName*.

--merge-file *MergeFile*
Specifies a file in the `openstack-config.conf` format that contains multiple values to be changed in a single operation. The properties in *MergeFile* can represent new properties or properties to be modified. If a section or property name in *MergeFile* begins with a '**-**' character, that section or property is deleted from the file. For example, a *MergeFile* with the following contents would delete the `ldap` section, set the connections property to 512, and delete the **noauth** property from the database section.

```
[-ldap]
[database]
connections = 512
-noauth =
```

manage
Performs management tasks on the object configuration.

Note: The object protocol is not supported in IBM Spectrum Scale 5.1.0.0. If you want to deploy object, upgrade to the IBM Spectrum Scale 5.1.0.1 or a later release.

--version-sync
After an upgrade of IBM Spectrum Scale, migrates the object configuration to be consistent with the level of installed packages.

Parameter common for both `mmobj config list` and `mmobj config change` commands:
--ccrfile *CCRFile*
Indicates the name of the Swift, Keystone, or Object configuration file stored in the CCR.

Some of the configuration files stored in the Cluster Configuration Repository (CCR) are:

- `account-server.conf`
- `container-reconciler.conf`
- `container-server.conf`
- `object-expirer.conf`
- `object-server.conf`
- `proxy-server.conf`
- `swift.conf`
- `keystone.conf`
- `keystone-paste.ini`

- `spectrum-scale-object.conf`
- `object-server-sof.conf`
- `spectrum-scale-objectizer.conf`

policy
Administers the storage policies for Object Storage:

list
Lists storage policies for Object Storage.

--policy-name *PolicyName*
Lists details of the specified storage policy, if it exists.

--policy-function *PolicyFunction*
Lists details of the storage policies with the specified function, if any exist.

--verbose
Lists the functions that are enabled for the storage policies.

Note: Use the **-Y** parameter to display the command output in a parseable format with a colon (:) as a field delimiter.

create
Creates a storage policy for Object Storage. The associated configuration files are updated and the ring files are created for the storage policy. The CES monitoring framework distributes the changes to the protocol nodes and restarts the associated services.

PolicyName
Specifies the name of the storage policy.

The policy name must be unique (case-insensitive), without spaces, and it must contain only letters, digits, or a dash.

-f *FilesetName*
Specifies the name of an existing fileset to be used for this storage policy. An existing fileset can be used provided it is not being used for an existing storage policy.

If no fileset name is specified with the command, the policy name is reused for the fileset with the prefix `obj_`.

--file-system *FilesystemName*
Specifies the name of the file system on which the fileset is created.

--i *MaxNumInodes*
Specifies the inode limit for the new inode space.

--enable-compression
Enables a compression policy. The Swift policy type is replication. If `--enable-compression` is used, `--compression-schedule` must be specified too and vice versa.

Every object stored within a container that is linked to this storage policy is compressed on a scheduled basis. This compression occurs as a background process. For object retrieval, no decompression is needed because it occurs automatically in the background.

--compression-schedule: *"MM:HH:dd:ww"*
Specifies the compression schedule if `--enable-compression` is used. The schedule must be given in this format, *MM:HH:dd:ww*:

MM = 0-59 minutes
Indicates the minute after the hour in which to run the job. The range is 0 - 59.

HH = 0-23
Indicates the hour in which to run the job. Hours are represented as numbers in the range 0 - 23.

dd = 1-31
Indicates the day of a month on which to run the job. Days are represented as numbers in the range 1 - 31.

ww = 0-7 (0=Sun, 7=Sun)
Indicates the days of the week on which to run the job. One or more values can be specified (comma-separated). Days are represented as numbers in the range 0 - 7. 0 and 7 represent Sunday. All days of a week are represented by *. This parameter is optional and the default is 0.

- Use * to specify every instance of a unit. For example, `dd = *` means that the job is scheduled to run every day.
- Comma-separated lists are allowed. For example, `dd = 1,3,5` means that the job is scheduled to run on every first, third, and fifth of a month.
- If `ww` and `dd` are both specified, the union is used.
- If you specify a range that uses `-`, it is not supported.
- Empty values are allowed for `dd` and `ww`. If empty, `dd` and/or `ww` are not considered.
- Empty values for `mm` and `hh` are treated as *.

--enable-encryption
Enables an encryption policy.

--enable-file-access
Enables a file-access policy. The file-access policies exist only in the region in which they were created. They do not support the multi-region capability. Objects stored within a container that is linked to this storage policy can be enabled for file protocol access.

--encryption-keyfile *EncryptionKeyFileName*

Specifies the encryption key file.

--force-rule-append

Specifies whether to append and establish the rule if other rules are existing.

Note: The enabled functions are displayed in the functions column of the `mmobj policy list` command output as follows:

- **`--enable-compression`** `compression`
- **`--enable-file-access`** `file-and-object-access`

change
Changes the state of the specified storage policy.

PolicyName
Specifies the name of the storage policy that needs to be changed.

--default
Sets the specified storage policy to be the default policy.

Note: You cannot set a deprecated storage policy as the default storage policy.

--deprecate *State*
Deprecates the specified storage policy. *State* can be either `yes` or `no`.

yes
Sets the state of the specified storage policy as deprecated.

no
Sets the state of the specified storage policy as not deprecated.

Note: You cannot deprecate the default storage policy.

--add-local-region

In a multi-region environment, adds the region of the current cluster to the specified storage policy. The associated fileset that is previously defined for the storage policy must exist or else it is created.

After the region is added, the multi-region configuration needs to be synced with the other regions by using the `mmobj multiregion` command.

Note: By default, a storage policy stores only objects in the region on which it was created. If the cluster is defined as multi-region, a storage policy can also be made multi-region by adding more regions to its definition.

--remove-region-number ***RegionNumber***

In a multi-region environment, removes a region from the specified storage policy. The associated fileset for the storage policy is not modified.

After the region is removed, the multi-region configuration needs to be synced with the other regions by using the `mmobj multiregion` command.

--compression-schedule: ***"MM:HH:dd:ww"***

Specifies the compression schedule if `--enable-compression` is used. The schedule must be given in this format, *MM:HH:dd:ww*:

MM = 0-59 minutes
Indicates the minute after the hour in which to run the job. The range is 0 - 59.

HH = 0-23
Indicates the hour in which to run the job. Hours are represented as numbers in the range 0 - 23.

dd = 1-31
Indicates the day of a month on which to run the job. Days are represented as numbers in the range 1 - 31.

ww = 0-7 (0=Sun, 7=Sun)
Indicates the days of the week on which to run the job. One or more values can be specified (comma-separated). Days are represented as numbers in the range 0 - 7. 0 and 7 represent Sunday. All days of a week are represented by `*`. This parameter is optional, and the default is 0.

- Use `*` to specify every instance of a unit. For example, `dd = *` means that the job is scheduled to run every day.
- Comma-separated lists are allowed. For example, `dd= 1,3,5` means that the job is scheduled to run on every first, third, and fifth of a month.
- If `ww` and `dd` are both specified, the union is used.
- If you specify a range that uses `-`, it is not supported.
- Empty values are allowed for `dd` and `ww`. If empty, `dd` and/or `ww` are not considered.
- Empty values for `mm` and `hh` are treated as `*`.

file-access

Manages file-access capability, `ibmobjectizer` service and object access for files (objectizes) in a unified file and object access environment.

enable

Enables the file-access capability and the `ibmobjectizer` service.

If the file access capability is already enabled and the `ibmobjectizer` service is stopped, this option starts the `ibmobjectizer` service.

disable

Disables the file-access capability and the `ibmobjectizer` service.

--objectizer
This option stops only the `ibmobjectizer` service, it does not disable the file-access capabilities.

objectize

Enables files for object access (objectizes) in a unified file and object access environment.

--object-path *ObjectPath*
The fully qualified path of a file or a directory that you want to enable for access through the object interface. If a fully qualified path to a directory is specified, the command enables all the files from that directory for access through the object interface. This parameter is mandatory for the `mmobj file-access` command if the **--storage-policy** parameter is not specified.

--storage-policy *PolicyName*
The name of the storage policy for which you want to enable files for the object interface. This is a mandatory parameter for the `mmobj file-access` command if the **--object-path** parameter is not specified. If only this parameter is specified, the command enables all files for object interface from the fileset that is associated with the specified storage policy.

--account-name *AcccountName*
The account name for which you want to enable files for access through the object interface. The `--storage-policy` parameter is mandatory if you are using this parameter.

--container-name *ContainerName*
The container name for which you want to enable files for access through the object interface. You must specify the **--storage-policy** and the **--account-name** with this parameter.

--object-name *ObjectName*
The object name for which you want to enable files for access through the object interface. You must specify **--storage-policy**, `--account-name`, and **--container-name** parameters with this parameter.

-N | --node *NodeName*
The node on which to run the command. This parameter is optional and if it is not specified, the command is run on the current node if it is a protocol node. If the current node is not a protocol node, an available protocol node is selected.

link-fileset

Enables object access to the specified fileset path.

--sourcefset-path *FilesetPath*
The fully qualified non-object fileset path that must be enabled for object access.

--account-name *AccountName*
The name of the swift account or the keystone project. The contents of the fileset are accessible from this account when it is accessed by using the object interface.

--container-name *ContainerName*
The name of the container. The contents of the fileset belong to this container when it is accessed by using the object interface.

--fileaccess-policy-name *PolicyName*
The name of the file-access enabled storage policy.

--update-listing
Updates the container database with the existing files in the provided fileset.

If the **--update-listing** option is used, then the `sourcefset-path` must be the exact fileset junction path and the fileset must be derived from the object file system.

multiregion
Administers multi-region object deployment. For more information on multi-region object deployment and capabilities, see *Overview of multi-region object deployment* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

list
Lists the information about the region.

Note: Use the **-Y** parameter to display the command output in a parseable format with a colon (:) as a field delimiter.

enable

Enables the cluster for multi-region support.

Only the first cluster of the region can run the enable command. Subsequent regions join the multi-region cluster during installation with the use of the **--join-region-file** flag of the `mmobj swift base` command.

export

Exports the multi-region configuration environment so that other regions can be installed into the multi-region cluster or other regions can be synced to this region.

If successful, a region checksum is printed in the output. This checksum can be used to ensure that different regions are in sync when the `mmobj multiregion import` command is run.

Note: When region-related information changes, such as CES IPs and storage policies, all regions must be updated with the changes.

--region-file *RegionFile*
Specifies a path to store the multi-region data.

This file is created.

import

Imports the specified multi-region configuration environment into this region.

If successful, a region checksum for this region is printed in the output. If the local region configuration matches the imported configuration, the checksums match. If they differ, then it means that some configuration information in the local region needs to be synced to the other regions. This can happen when a configuration change in the local region, such as adding CES IPs or storage policies, is not yet synced with the other regions. If so, the multi-region configuration for the local region needs to be exported and synced to the other regions.

--region-file *RegionFile*
Specifies the path to a multi-region data file created by using the `mmobj multiregion export` command.

remove

Completely removes a region from the multi-region environment. The removed region is no longer accessible by other regions.

After the region is removed, the remaining regions need to have their multi-region information that is synced with this change by using the `mmobj multiregion export` and `mmobj multiregion import` commands.

--region-number *RegionNumber*
Specifies the region number that you need to remove from the multi-region configuration.

--force
Indicates that all the configuration information for the specified region needs to be permanently deleted.

s3

Enables and disables the S3 API without manually changing the configuration.

enable
Enables the S3 API.

disable
Disables the S3 API.

list
Verifies whether the S3 API is enabled or disabled.

Note: Use the **-Y** parameter to display the command output in a parseable format with a colon (:) as a field delimiter.

-Y
Displays headers and output in a machine-readable and colon-delimited format.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

Exit status

0
Successful completion.

nonzero
A failure occurs.

Security

You must have root authority to run the `mmobj` command.

The node on which the command is run must be able to run remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To specify configuration of Object protocol service with local keystone and S3 enabled, run the following command:

```
# mmobj swift base  -g /gpfs/ObjectFS --cluster-hostname cluster-ces-ip.ibm --local-
keystone
--enable-s3 --pwd-file PasswordFile
```

   A sample output is as follows:

```
mmobj swift base: Creating fileset /dev/ObjectFS object_fileset
mmobj swift base: Configuring Keystone server in /gpfs/cesshared/ces/object/keystone
mmobj swift base: Configuring Swift services
Configuration complete
```

2. To list object configuration settings for `proxy-server.conf`, DEFAULT section, run the following command:

```
# mmobj config list --ccrfile proxy-server.conf --section DEFAULT
```

   A sample output is as follows:

```
[DEFAULT]
bind_port = 8080
workers = auto
user = swift
log_level = ERROR
```

3. To change the number of worker processes that each object server starts, update the `object-server.conf` file as shown here:

```
# mmobj config change --ccrfile object-server.conf --section DEFAULT --property workers
--value 16
```

4. To change the configuration value of `paste.filter_factory` that is in the `filter:s3_extension` section of the `keystone-paste.ini` configuration file, run the following command:

```
# mmobj config change --ccrfile keystone-paste.ini --section filter:s3_extension
--property paste.filter_factory --value keystone.contrib.s3:S3Extension.factory
```

5. To migrate the configuration data after an upgrade of IBM Spectrum Scale, run the following command:

```
# mmobj config manage --version-sync
```

A sample output is as follows:

```
mmobj config manage: Checking system state
mmobj config manage: Processing Keystone
mmobj config manage: Processing spectrum-scale-object.conf
mmobj config manage: Processing object-server-sof.conf
mmobj config manage: Processing spectrum-scale-objectizer.conf
mmobj config manage: Processing object-server.conf
mmobj config manage: Processing wsgi-keystone.conf
mmobj config manage: Processing proxy-server.conf
mmobj config manage: Processing keystone.conf
mmobj config manage: Migration of configuration is complete.

mmobj config manage: gpfs.base@5.0.3
mmobj config manage: spectrum-scale-object@5.0.3
mmobj config manage: Processing keystone-paste.ini
```

6. To create a new storage policy `CompressionTest` with the compression function enabled and with the compression schedule specified, run the following command:

```
# mmobj policy create CompressionTest --enable-compression
--compression-schedule '20:5,11,17,23:*:*'
```

A sample output is as follows:

```
[I] Getting latest configuration from ccr
[I] Creating fileset /dev/ObjectFS:obj_CompressionTest
[I] Creating new unique index and build the object rings
[I] Updating the configuration
[I] Uploading the changed configuration
```

This compression schedule indicates that the fileset that is associated with the policy is compressed at the 20th minute of the 5th, 11th, 17th, and 23rd hour of the day on every day of every week.

7. To list storage policies for Object Storage with details of functions available with those storage policies, run the following command:

```
# mmobj policy list --verbose
```

A sample output is as follows:

```
Index       Name          Deprecated  Fileset          Fileset Path                        Functions               Function Details
File System
-----------------------------------------------------------------------------------------------------------------------------------
----------
0           SwiftDefault              object_fileset   /ibm/ObjectFS/object_fileset                                                  ObjectFS
11751509160 sof-policy                obj_sof-policy   /ibm/ObjectFS/obj_sof-policy   file-and-object-access   regions="1"        ObjectFS
11751509230 mysofpolicy               obj_mysofpolicy  /ibm/ObjectFS/obj_mysofpolicy  file-and-object-access   regions="1"        ObjectFS
11751510260 Test19                    obj_Test19       /ibm/ObjectFS/obj_Test19                                regions="1"        ObjectFS
```

8. To change the default storage policy, run the following command:

```
# mmobj policy change sof-policy --default
```

The system displays `sof-policy` as the default storage policy.

9. To enable object access for an account, run the following command:

```
# mmobj file-access --storage-policy sof_policy --account-name admin
```

A sample output is as follows:

```
Loading objectization configuration from CCR
Fetching storage policy details
Creating container to database map
Performing objectization
Objectization complete
```

10. To enable object access for a container, run the following command:

```
# mmobj file-access objectize --storage-policy sof_policy
--account-name admin --container-name container1
```

11. To enable object access on a file when you specify a storage policy, run the following command:

```
# mmobj file-access objectize --storage-policy sof_policy
--account-name admin --container-name container1 --object-name file1.txt
```

12. To enable object access on a file, run the following command:

```
# mmobj file-access objectize --object-path
/ibm/ObjectFS/obj_sofpolicy1/s69931509221z1device1/AUTH_12345/container1/file1.txt
```

13. To list the information about a region, run the following command:

```
# mmobj multiregion list
```

A sample output is as follows:

```
Region  Endpoint   Cluster Name     Cluster Id
------  ---------  ---------------  -------------------
1       RegionOne  europe.gpfs.net  3694106483743716196
2       Region2    asia.gpfs.net    1860802811592373112
```

14. To set up the initial multi-region environment on the first region, run the following command:

```
# mmobj multiregion enable
```

A sample output is as follows:

```
mmobj multiregion: Multi-region support is enabled in this cluster.
Region number: 1
```

15. To export multi-region data for use by other clusters to join multi-region, run the following command:

```
# mmobj multiregion export --region-file /tmp/region2.dat
```

A sample output is as follows:

```
mmobj multiregion: The Object multi-region export file
was successfully created: /tmp/region2.dat
mmobj multiregion: Region checksum is: 34632-44791
```

16. To import the specified multi-region configuration environment created by the export command into a region, run the following command:

```
# mmobj multiregion import --region-file /tmp/region2.dat
```

A sample output is as follows:

```
mmobj multiregion: The region config has been updated.
mmobj multiregion: Region checksum is: 34632-44791
```

17. To remove a region that is designated by region number 2 from a multi-region environment and to remove all configuration information of the specified region, run the following command:

```
# mmobj multiregion remove --remove-region-number 2 --force
```

A sample output is as follows:

```
mmobj multiregion: Permanently removing region 2
(asia.gpfs.net 1860802811592373112) from multi-region configuration.
mmobj multiregion: Updating ring files.
mmobj multiregion: Successfully removed region 2.
Object services on region 2 will need to be unconfigured and its endpoint
removed from Keystone.
```

18. To create a policy with no force add where only the default GPFS policy rule is established, run the following command:

```
# mmobj policy create enc-policy-1 --enable-encryption
--encryption-keyfile /root/enc/enc.key
```

The system creates the policy, adds and establishes the rule within the GPFS policy rules, and displays the following output:

```
[I] Getting latest configuration from ccr
[I] Creating fileset /dev/ObjectFS:obj_enc-policy-1
[I] Creating GPFS Encryption Rule
[I]  The following new encryption rule has been stored to
file:/var/mmfs/ces/policyencryption.rule
and is established within the  GPFS policies.

/* begin of the encryption rule for fileset obj_enc-policy-1 */
rule 'enc-1_enc-policy-1' set encryption 'encryption_enc-policy-1'
for fileset('obj_enc-policy-1')
where name like '%'

rule 'enc-2_enc-policy-1' encryption 'encryption_enc-policy-1'
is  ALGO 'DEFAULTNISTSP800131A'
KEYS('KEY-2712784a-d1ee-4c86-bf97-b88f918cbd12:sklm')
/* end of the encryption rule for fileset obj_enc-policy-1 */

[I] Creating new unique index and building the object rings
[I] Updating the configuration
[I] Uploading the changed configuration
```

19. To create a policy with force add, run the following command:

```
# mmobj policy create enc-policy-2 --enable-encryption
--encryption-keyfile /root/enc/enc.key
--force-rule-add
```

The system creates the policy, adds and establishes the rule within GPFS policy rules, and displays the following output:

```
[I] Getting latest configuration from ccr
[I] Creating fileset /dev/ObjectFS:obj_enc-policy-2
[I] Creating GPFS Encryption Rule
[I]  The following new encryption rule has been stored to file:/var/mmfs/ces/
policyencryption.rule
 and is established within the  GPFS policies.

/* begin of the encryption rule for fileset obj_enc-policy-2 */
rule 'enc-1_enc-policy-2' set encryption 'encryption_enc-policy-2'
for fileset('obj_enc-policy-2')
where name like '%'

rule 'enc-2_enc-policy-2' encryption 'encryption_enc-policy-2' is
ALGO 'DEFAULTNISTSP800131A'
KEYS('KEY-2712784a-d1ee-4c86-bf97-b88f918cbd12:sklm')
/* end of the encryption rule for fileset obj_enc-policy-2 */

[I] Creating new unique index and building the object rings
[I] Updating the configuration
[I] Uploading the changed configuration
```

20. To create a policy with no force add, run the following command:

```
# mmobj policy create enc-policy-3 --enable-encryption
--encryption-keyfile /root/enc/enc.key
```

The system creates the policy, adds the rule but does not establish the rule within the GPFS policy rules, and displays the following output:

```
[I] Getting latest configuration from ccr
[I] Creating fileset /dev/ObjectFS:obj_enc-policy-3
[I] Creating GPFS Encryption Rule
[I]  The following new encryption rule has been stored to
file:/var/mmfs/ces/policyencryption.rule
 but is not established within  the GPFS policies.
```

```
/* begin of the encryption rule for fileset obj_enc-policy-3 */
rule 'enc-1_enc-policy-3' set encryption 'encryption_enc-policy-3'
for fileset('obj_enc-policy-3')
where name like '%'

rule 'enc-2_enc-policy-3' encryption 'encryption_enc-policy-3' is
ALGO 'DEFAULTNISTSP800131A'
KEYS('KEY-2712784a-d1ee-4c86-bf97-b88f918cbd12:sklm')
/* end of the encryption rule for fileset obj_enc-policy-3 */

[I] Creating new unique index and building the object rings
[I] Updating the configuration
[I] Uploading the changed configuration
```

21. To enable the `ibmobjectizer` service, run the following command:

```
# mmobj file-access enable
```

The system enables the file-access capability and starts the `ibmobjectizer` service.

22. To disable the `ibmobjectizer` service, run the following command:

```
# mmobj file-access disable
```

The system disables the file-access capability and stops the `ibmobjectizer` service.

23. To use the disable **`--objectizer-daemon`** parameter, run the following command:

```
# mmobj file-access disable --objectizer-daemon
```

The system disables the `ibmobjectizer` service.

24. To use the object-path parameter, run the following command:

```
# mmobj file-access objectize --object-path
/ibm/ObjectFS/fileset1/Auth_12345/container1/file1.txt
```

The system objectizes `file1.txt` at `/ibm/ObjectFS/fileset1/Auth_12345/container1/` and enables it for access through the object interface:

```
Loading objectization configuration from CCR
Fetching storage policy details
Performing objectization
Objectization complete
```

25. To use the storage-policy parameter for file objectization, run the following command:

```
# mmobj file-access objectize --storage-policy sof_policy --account-name admin
 --container-name container1 --object-name file1.txt
```

The system objectizes `file1.txt` from the container named `container1`:

```
Loading objectization configuration from CCR
Fetching storage policy details
Performing objectization
Objectization complete
```

26. To use the node parameter for running the command on a remote node, run the following command:

```
# mmobj file-access objectize --node gpfs_node1
--storage-policy sof_policy --account-name admin
```

The command is run on the `gpfs_node1` node and all the files from all containers within the admin account are objectized. If **`--node`** or **`-N`** is not specified, the `mmobj file-access objectize` command checks if the current node is CES node or not. If the current node is a CES node, the command is run on the current node. If the current node is not a CES node, the command is run on a random remote CES node:

```
Loading objectization configuration from CCR
Fetching storage policy details
```

```
Performing objectization
Objectization complete
```

See also

- “mmces command” on page 133
- “mmchconfig command” on page 170
- “mmlscluster command” on page 492
- “mmlsconfig command” on page 495
- “mmnfs command” on page 560
- “mmsmb command” on page 709
- “mmuserauth command” on page 738

Location

/usr/lpp/mmfs/bin

mmperfmon command

Configures the Performance Monitoring tool and lists the performance metrics.

Synopsis

```
mmperfmon config generate --collectors CollectorNode[,CollectorNode...]
                          [ --config-file InputFile ]
```

or

```
mmperfmon config add --sensors SensorFile
```

or

```
mmperfmon config add --apikey key_name
```

or

```
mmperfmon config update { [--collectors CollectorNode[,CollectorNode...] ]
                          [ --config-file InputFile ] [ Attribute=value ... ] }
```

or

```
mmperfmon config update --apikey key_name
```

or

```
mmperfmon config delete {--all |--sensors  Sensor[,Sensor...] }
```

or

```
mmperfmon config delete --apikey key_name
```

or

```
mmperfmon config show [--config-file OutputFile]
```

or

```
mmperfmon config show --apikey  [key_name | all]
```

or

```
mmperfmon query  Metric[,Metric...] |  Key[,Key...] | NamedQuery
                [StartTime EndTime | Duration] [Options]
```

or

```
mmperfmon query compareNodes ComparisonMetric [StartTime EndTime | Duration] [Options]
```

or

```
mmperfmon report top [StartTime EndTime | Duration][Options]
```

or

```
mmperfmon delete {--expiredKeys |--key Key[,Key...]}
```

Availability

Available on all IBM Spectrum Scale editions.

The protocol functions provided in this command, or any similar command, are generally referred to as CES (Cluster Export Services). For example, protocol node and CES node are functionally equivalent terms.

Description

The **mmperfmon config** command modifies the performance monitoring tool by updating the configuration that is stored in IBM Spectrum Scale. It can be used to generate an initial configuration to update reporting periods of different sensors, or to restrict sensors to a defined set of nodes.

The **mmperfmon config** command is used to query metrics in a cluster from the performance metrics collector. Output can be delivered either in a raw format, formatted table layout or as a CSV export.

In addition to metrics known by the performance collector, the **mmperfmon config** command can also run predefined named queries or use predefined computed metrics. You can specify a bucket size for each record to return in number of seconds and the number of buckets to retrieve. You can also specify the duration or a time range for which the query can run.

When the performance monitoring tool is being used in a container environment, the transient network interfaces and file system mount points can increase the amount of metadata or keys within the tool and slowdown the metric queries. To prevent this issue, use the filter attribute of the Network and Disk Free sensors. Usage of the **mmperfmon** command to set the filter conditions is shown in Example 5.

Parameters

config

generate

Generates the configuration of the performance monitoring tool.

Note: After the configuration is generated, do not forget to turn on monitoring through the **mmchnode** command.

--collectors *CollectorNode*[,*CollectorNode...*] specifies the set of collectors to which the sensors report their performance measurements. The number of collectors that each sensor reports to can be specified through **colRedundancy** parameter in the template sensor configuration file (see --config-file). Federated collectors are automatically configured between these collectors. For more information, see *Configuring multiple collectors* section in the *IBM Spectrum Scale: Problem Determination Guide*.

--config-file *InputFile* specifies the template sensor configuration file to use. If this option is not provided, the `/opt/IBM/zimon/defaults/ZIMonSensors.cfg` file is used.

add

Adds a new sensor to the performance monitoring tool.

--sensors *SensorFile* adds the sensors that are specified in *SensorFile* to the sensor configuration. Multiple sensors in the configuration file need to be separated by a comma. Following is a sample *SensorFile*:

```
sensors = {
name = "MySensor"
# sensor disabled by default
period = 0
type = "Generic"
}
```

The generic sensor and a sensor-specific configuration file need to be installed on all the nodes where the generic sensor is to be activated.

--apikey *key_name* creates an API key entry for a specific `key_name`, and stores the information in CCR, so that all the cluster nodes are aware of it. If the `key_name` already exists, an error message is displayed.

Note: The `--apikey` flag is mutually exclusive with other flags. The default API key *scale_default* is system-defined, and is generated automatically.

update
Updates the existing configuration.

--collectors *CollectorNode*[,*CollectorNode...*] updates the collectors to be used by the sensors and for federation (see **`config generate`** for details).

--config-file *InputFile* specifies a template sensor configuration file to use. This file overwrites the currently used configuration with the configuration that is specified in *InputFile*.

Attribute=value ... specifies a list of attribute value assignments. This sets the value of attribute *Attribute* to *value*. *Attribute* is a combination of a sensor name and one of its parameters that are separated by a dot as shown: *<sensor>.<parameter>*. For example, `CPU.period`.

The following attributes are supported:

<sensor>.period
Specifies the seconds between each invocation of the sensor. This parameter is supported for all sensors.

<sensor>.restrict
Specifies a node or node class. The invocation of the sensor is limited to the nodes specified. This parameter is supported for all sensors.

<sensor>.filter
Specifies a sensor-specific element to be ignored when the sensor retrieves data. This parameter is supported for sensors Network and DiskFree.

--apikey *key_name* updates an existing key entry for a given `key_name`, and stores the information in CCR. A new key value is generated and replaces the previous one. If the *key_name* does not exist, then an error message is displayed.

Note: The `--apikey` flag is mutually exclusive with other flags.

delete
Removes configuration of the performance monitoring tool or the specified sensors.

--sensors *Sensor*[,*Sensor...*] removes the sensors with the specified names from the performance monitoring configuration.

--all removes the entire performance monitoring configuration from IBM Spectrum Scale.

--apikey *key_name* deletes an existing key entry for a given `key_name`, and stores the information in CCR. If the *key_name* does not exist, an error message is displayed.

Note: The `--apikey` flag is mutually exclusive with other flags. The default API key *scale_default* is system-defined, and cannot be deleted.

show
Displays the currently active performance monitoring configuration. Specifies the following options:

--config-file *OutputFile* specifies that the output is saved to the *OutputFile*.

--apikey *key_name* shows existing key entries. If `key_name` is specified, then only the corresponding data is shown. If the *key_name* does not exist, then an error message is shown. If no argument or `all` is given, then all entries are displayed.

Note: The `--apikey` flag is mutually exclusive with other flags.

query

Metric[,*Metric...*] specifies a comma-separated list of metrics for displaying in the output.

Key[,*Key...*] specifies a key that can consist of a node name, sensor group, or optional filters, and metrics that are separated by the pipe symbol (|). For example,

```
"cluster1.ibm.com|CTDBStats|locking|db_hop_count_bucket_00"
```

NamedQuery specifies the name of a predefined query.

compareNodes compares the specified metrics for all nodes in the system. The query creates one column per existing node and only one metric can be compared.

ComparisonMetric specifies the name of the metric to be compared when using the **compareNodes** query.

StartTime specifies the start timestamp for query in the YYYY-MM-DD-hh:mm:ss format.

EndTime specifies the end timestamp for query in the YYYY-MM-DD-hh:mm:ss format. If it is not specified, the query returns results until the present time.

Duration specifies the number of seconds into the past from present time or *EndTime*.

Options specifies the following options:

- -N or --Node *NODENAME* specifies the node from where the metrics are retrieved.

  For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.
- --bucket-size *BUCKET_SIZE* specifies the bucket size (number of seconds), default is 1.
- --number-buckets *NUMBER_BUCKETS* specifies the number of buckets (records) to display, default is 10.
- --filter *FILTER* specifies the filter criteria for the query to run. To see the list of filters in the node use the **mmperfmon query --list filters** command.
- --format *FORMAT* specifies a common format for all columns.
- --csv provides the output in the CSV format.
- --raw provides the output in a raw format rather than a tabular format.
- --short displays the column names in a short form when there are too many to fit into a row.
- --nice displays the column headers in the output in a bold and underlined typeface.
- --resolve displays the resolved computed metrics and metrics that are used.
- --list {computed | metrics | keys | filters | queries | expiredKeys | all} lists the following information:
  - *computed* displays the computed metrics.
  - *metrics* displays the metrics.
  - *keys* lists the keys.
  - *filters* lists the filters.
  - *queries* lists the available predefined queries.
  - *expiredKeys* lists the group keys for the entities that do not return any metrics values within the default retention period of 14 days.
  - *all* displays the computed metrics, metrics, keys, filters, and queries.

report

Returns a report.

top

Returns a report of the top processes by the CPU.

StartTime specifies the start timestamp for query in the YYYY-MM-DD-hh:mm:ss format.

EndTime specifies the end timestamp for a query in the YYYY-MM-DD-hh:mm:ss format. If it is not specified, the query returns results until the present time.

Duration specifies the number of seconds into the past from the present time or *EndTime*.

Options specifies the following options:

- -N or --Node *NODENAME* specifies the node from where the metrics are retrieved.

  For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.
- --bucket-size *BUCKET_SIZE* specifies the bucket size in number of seconds. Default is 1.
- --number-buckets *NUMBER_BUCKETS* specifies the number of buckets to display. Default is 10.
- --json provides the output in a JSON format.
- --raw provides the output in a RAW format.

  Note: The --json and --raw options display similar outputs.
- --filter *FILTER* specifies the filter criteria for the query to run. To see the list of filters in the node use the **mmperfmon query --list filters** command. Currently, the only supported filter is node.

delete

Removes expired keys from the performance monitoring tool database.

--key *Key*[,*Key...*] specifies the key or list of keys that must be removed from the performance monitoring tool database, if they are expired keys. The keys are displayed as a comma-separated string.

--expiredKeys specifies all the expired keys must be removed from the performance monitoring tool database.

Note: A group key is the part of the metric key string that represents a base entity. For example, for the keys:

```
gpfsgui-cluster-1.novalocal|GPFSInodeCap|nfs_shareFS|gpfs_fs_inode_alloc
gpfsgui-cluster-1.novalocal|GPFSInodeCap|nfs_shareFS|gpfs_fs_inode_free
gpfsgui-cluster-1.novalocal|GPFSInodeCap|nfs_shareFS|gpfs_fs_inode_max
gpfsgui-cluster-1.novalocal|GPFSInodeCap|nfs_shareFS|gpfs_fs_inode_used
```

the group key would be `gpfsgui-cluster-1.novalocal|GPFSInodeCap|nfs_shareFS`. Expired keys are group keys, which are detected by IBM Spectrum Scale monitoring tool. However, these group keys are not found in the current cluster configuration and do not return metrics for the default retention period of at least 14 days.

Exit status

0

Successful completion.

1

Invalid arguments given

2

Invalid option

3

No node found with a running performance collector

4

Performance collector backend signaled bad query, for example, `no data for this query`.

Security

You must have root authority to run the `mmperfmon` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. The

performance monitoring tool uses the GPFS™ cluster daemon node names and network to communicate between nodes. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To generate configuration for the c89f8v03 collector node, issue the following command:

```
# mmperfmon config generate --collectors c89f8v03
```

   A sample output is as follows:

```
mmperfmon: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
Tue Oct 27 20:40:07 EDT 2015: mmcommon pushSdr_async: mmsdrfs propagation started
```

2. To add /tmp/SensorFile sensor to the performance monitoring tool, issue the following command:

```
# mmperfmon config add --sensors /tmp/SensorFile
```

   A sample output is as follows:

```
mmperfmon: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
Tue Oct 27 20:44:33 EDT 2015: mmcommon pushSdr_async: mmsdrfs propagation started
# mmperfmon config show | tail -12
{
        name = "NFSIO"
        period = 0
        proxyCmd = "/opt/IBM/zimon/GaneshaProxy"
        restrict = "cesNodes"
        type = "Generic"
},
{
        name = "TestAdd"
        period = 4
}
smbstat = ""
```

3. To create a new API key for the following cases:

   a. If no keys are defined yet, then issue the following command:

```
# mmperfmon config add --apikey scale_default
```

      A sample output is as follows:

```
File _perfmon.keys does not exist in CCR
Create a new empty API key Json file to be uploaded to CCR
API key file _perfmon.keys successfully uploaded to CCR
[root@gpfs-21 ~]#


[root@gpfs-21 ~]# mmperfmon config show --apikey scale_default
{
    "key": "aedc2798-e2b0-4cda-8f12-da1e778c1504",
    "comment": "user comment"
}
[root@gpfs-21 ~]#
```

   b. If keys are already defined, then issue the following command:

```
# mmperfmon config add --apikey scale_user1
```

      A sample output is as follows:

```
API key file _perfmon.keys successfully uploaded to CCR
[root@node-11 ~]#
```

4. To update the NFSIO.period value to 5, issue the following command:

```
# mmperfmon config update NFSIO.period=5
```

A sample output is as follows:

```
mmperfmon: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
Tue Oct 27 20:47:53 EDT 2015: mmcommon pushSdr_async: mmsdrfs propagation started

# mmperfmon config show | tail -9
},
{
        name = "NFSIO"
        period = 5
        proxyCmd = "/opt/IBM/zimon/GaneshaProxy"
        restrict = "cesNodes"
        type = "Generic"
}
smbstat = ""
```

5. To update the Network.filter value to ignore multiple network interfaces, issue the following command:

```
# mmperfmon config update Network.filter="netdev_name=veth.*|docker.*|flannel.*|cali.*|
cbr.*"
```

A sample output is as follows:

```
mmperfmon: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
Tue Jun 23 17:25:45 CEST 2020: mmcommon pushSdr_async: mmsdrfs propagation started
Tue Jun 23 17:25:51 CEST 2020: mmcommon pushSdr_async: mmsdrfs propagation completed; mmdsh
rc=0
```

6. To update an API key value, issue the following command:

```
# mmperfmon config update --apikey animal
```

A sample output is as follows:

```
API key file _perfmon.keys successfully uploaded to CCR
[root@gpfs-21 ~]#
```

7. To remove the TestAdd sensor, issue the following command:

```
# mmperfmon config delete --sensors TestAdd
```

A sample output is as follows:

```
mmperfmon: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
Tue Oct 27 20:46:23 EDT 2015: mmcommon pushSdr_async: mmsdrfs propagation started
Tue Oct 27 20:46:28 EDT 2015: mmcommon pushSdr_async: mmsdrfs propagation completed; mmdsh rc=0

# mmperfmon config show | tail -12
{
        name = "GPFSDiskCap"
        period = 0
},
{
        name = "NFSIO"
        period = 0
        proxyCmd = "/opt/IBM/zimon/GaneshaProxy"
        restrict = "cesNodes"
        type = "Generic"
}
smbstat = ""
```

8. To display the currently active performance monitoring configuration, issue the following command:

```
# mmperfmon config show
```

A sample output is as follows:

```
cephMon = "/opt/IBM/zimon/CephMonProxy"
cephRados = "/opt/IBM/zimon/CephRadosProxy"
```

```
colCandidates = "c89f8v03"
colRedundancy = 1
collectors = {
        host = ""
        port = "4739"
}
config = "/opt/IBM/zimon/ZIMonSensors.cfg"
ctdbstat = ""
daemonize = T
hostname = ""
ipfixinterface = "0.0.0.0"
logfile = "/var/log/zimon/ZIMonSensors.log"
loglevel = "info"
mmcmd = "/opt/IBM/zimon/MMCmdProxy"
mmdfcmd = "/opt/IBM/zimon/MMDFProxy"
mmpmon = "/opt/IBM/zimon/MmpmonSockProxy"
piddir = "/var/run"
release = "4.2.0-0"
sensors = {
        name = "CPU"
        period = 1
},
{
        name = "Load"
        period = 1
},
{
        name = "Memory"
        period = 1
},
{
        name = "Network"
        period = 1
},
{
        name = "Netstat"
        period = 0
},
{
        name = "Diskstat"
        period = 0
},
{
        name = "DiskFree"
        period = 600
},
{
        name = "GPFSDisk"
        period = 0
},
{
        name = "GPFSFilesystem"
        period = 1
},
{
        name = "GPFSNSDDisk"
        period = 1
        restrict = "nsdNodes"
},
{
        name = "GPFSPoolIO"
        period = 0
},
{
        name = "GPFSVFS"
        period = 1
},
{
        name = "GPFSIOC"
        period = 0
},
{
        name = "GPFSVIO"
        period = 0
},
{
        name = "GPFSPDDisk"
        period = 1
        restrict = "nsdNodes"
},
{
        name = "GPFSvFLUSH"
```

```
        period = 0
},
{
        name = "GPFSNode"
        period = 1
},
{
        name = "GPFSNodeAPI"
        period = 1
},
{
        name = "GPFSFilesystemAPI"
        period = 1
},
{
        name = "GPFSLROC"
        period = 0
},
{
        name = "GPFSCHMS"
        period = 0
},
{
        name = "GPFSAFM"
        period = 0
},
{
        name = "GPFSAFMFS"
        period = 0
},
{
        name = "GPFSAFMFSET"
        period = 0
},
{
        name = "GPFSRPCS"
        period = 0
},
{
        name = "GPFSFilesetQuota"
        period = 3600
},
{
        name = "GPFSDiskCap"
        period = 0
},
{
        name = "NFSIO"
        period = 0
        proxyCmd = "/opt/IBM/zimon/GaneshaProxy"
        restrict = "cesNodes"
        type = "Generic"
},
{
        name = "SwiftAccount"
        period = 1
        restrict = "cesNodes"
        type = "generic"
},
{
        name = "SwiftContainer"
        period = 1
        restrict = "cesNodes"
        type = "generic"
},
{
        name = "SwiftObject"
        period = 1
        restrict = "cesNodes"
        type = "generic"
},
{
        name = "SwiftProxy"
        period = 1
        restrict = "cesNodes"
        type = "generic"
}
smbstat = ""
```

9. To show all the API key values, issue the following command:

```
# mmperfmon config show --apikey all
```

A sample output is as follows:

```
{
    "scale_default": {
        "key": "aedc2798-e2b0-4cda-8f12-da1e778c1504",
        "comment": "user comment"
    },
    "forrest": {
        "key": "c07abeb9-b137-4074-95c7-e49eaf92dfce",
        "comment": "user comment"
    },
    "buttermilk": {
        "key": "f2f08be9-c743-4626-bde2-99e05502732c",
        "comment": "user comment"
    },
    "honey": {
        "key": "7d54d052-2600-405c-b316-8b36a88b95ce",
        "comment": "user comment"
    },
    "animal": {
        "key": "ab2e3a71-d869-4e4e-a690-68de58ecbc6e",
        "comment": "user comment"
    }
}
[root@gpfs-21 ~]#
```

10. To list metrics by key, for given node, sensor group and metrics, issue the following command:

```
# mmperfmon query "cluster1.ibm.com|CTDBDBStats|locking|db_hop_count_bucket_00"
```

A sample output is as follows:

```
Row           Timestamp db_hop_count_bucket_00
  1 2015-04-08-12:54:53                     0
  2 2015-04-08-12:54:54                     0
  3 2015-04-08-12:54:55                     0
  4 2015-04-08-12:54:56                     0
  5 2015-04-08-12:54:57                     0
  6 2015-04-08-12:54:58                     0
  7 2015-04-08-12:54:59                     0
  8 2015-04-08-12:55:00                     0
  9 2015-04-08-12:55:01                     0
 10 2015-04-08-12:55:02                     0
```

11. To list the two metrics `nfs_read_lat` and `nfs_write_lat` for a specific time range, filtered by an export and NFS version with 60 seconds buckets (one record represents 60 seconds), issue the following command:

```
# mmperfmon query nfs_read_lat,nfs_write_lat 2014-12-19-11:15:00 2014-12-19-11:20:00 --filter export=/ibm/gpfs/nfsexport,nfs_ver=NFSv3 -b 60
```

A sample output is as follows:

```
Row           Timestamp nfs_read_lat nfs_write_lat
  1 2015-04-10-09:24:00            0             0
  2 2015-04-10-09:24:10            0             0
  3 2015-04-10-09:24:20            0             0
  4 2015-04-10-09:24:30            0             0
  5 2015-04-10-09:24:40            0             0
  6 2015-04-10-09:24:50            0             0
  7 2015-04-10-09:25:00            0             0
  8 2015-04-10-09:25:10            0             0
  9 2015-04-10-09:25:20            0             0
 10 2015-04-10-09:25:30            0             0
 11 2015-04-10-09:25:40            0             0
 12 2015-04-10-09:25:50     45025738    1882453623
 13 2015-04-10-09:26:00            0             0
 14 2015-04-10-09:26:10            0             0
 15 2015-04-10-09:26:20            0             0
 16 2015-04-10-09:26:30            0             0
 17 2015-04-10-09:26:40            0             0
 18 2015-04-10-09:26:50            0             0
```

12. To list all available filters, issue the following command:

```
# mmperfmon query --list filters
```

A sample output is as follows:

```
Available Filters:

node
        gpfs-21.localnet.com
        gpfs-22.localnet.com
protocol
        smb2
db_name
        account_policy
        autorid
        brlock
        ctdb
        dbwrap_watchers
        g_lock
        group_mapping
        leases
        locking
        netlogon_creds_cli
        notify_index
        passdb
        registry
        secrets
        serverid
        share_info
        smbXsrv_open_global
        smbXsrv_session_global
        smbXsrv_tcon_global
        smbXsrv_version_global
gpfs_fs_name
        fs0
        gpfs0
gpfs_cluster_name
        gpfs-cluster-2.localnet.com
mountPoint
        /
        /boot
        /dev
        /dev/shm
        /gpfs/fs0
        /mnt/gpfs0
        /run
        /sys/fs/cgroup
operation
        break
        cancel
        close
        create
        find
        flush
        getinfo
        ioctl
        keepalive
        lock
        logoff
        negprot
        notify
        read
        sesssetup
        setinfo
        tcon
        tdis
        write
sensor
        CPU
        CTDBDBStats
        CTDBStats
        DiskFree
        GPFSFilesystemAPI
        GPFSVFS
        Load
        Memory
        Network
        SMBGlobalStats
        SMBStats
netdev_name
```

```
        eth0
        lo
```

13. To run a named query for export /ibm/gpfs/nfsexport and **nfs_ver NFSv3**, using default bucket size of 1 second, showing last 10 buckets, issue the following command:

```
# mmperfmon query nfsIORate --filter export=/ibm/gpfs/
nfsexport,nfs_ver=NFSv3,node=cluster1.ibm.com
```

A sample output is as follows:

```
Legend:
1:      cluster1.ibm.com|NFSIO|/ibm/gpfs/nfsexport|NFSv3|nfs_read_ops
2:      cluster2.ibm.com|NFSIO|/ibm/gpfs/nfsexport|NFSv3|nfs_write_ops

Row           Timestamp nfs_read_ops nfs_write_ops
  1 2015-05-11-13:32:57            0             0
  2 2015-05-11-13:32:58           90            90
  3 2015-05-11-13:32:59           90            90
  4 2015-05-11-13:33:00           90            91
  5 2015-05-11-13:33:01           91            90
  6 2015-05-11-13:33:02           91            92
  7 2015-05-11-13:33:03           89            88
  8 2015-05-11-13:33:04           91            92
  9 2015-05-11-13:33:05           93            92
 10 2015-05-11-13:33:06           89            89
```

14. To run a named query for export /ibm/gpfs/nfsexport and **nfs_ver NFSv3**, using bucket size of 1 minute, showing last 20 buckets (= 20 minutes), issue the following command:

```
# mmperfmon query nfsIORate --filter export=/ibm/gpfs/nfsexport,nfs_ver=NFSv3,node=cluster1.ibm.com -n 20 -b 60
```

A sample output is as follows:

```
Legend:
1:      cluster1.ibm.com|NFSIO|/ibm/gpfs/nfsexport|NFSv3|nfs_read_ops
2:      cluster2.ibm.com|NFSIO|/ibm/gpfs/nfsexport|NFSv3|nfs_write_ops

Row           Timestamp nfs_read_ops nfs_write_ops
  1 2015-05-11-13:31:00            0             0
  2 2015-05-11-13:32:00          280           280
  3 2015-05-11-13:33:00          820           820
  4 2015-05-11-13:34:00            0             0
  5 2015-05-11-13:35:00            0             0
  6 2015-05-11-13:36:00            0             0
  7 2015-05-11-13:37:00            0             0
  8 2015-05-11-13:38:00            0             0
  9 2015-05-11-13:39:00         1000          1000
 10 2015-05-11-13:40:00         1000          1000
 11 2015-05-11-13:41:00            0             0
 12 2015-05-11-13:42:00            0             0
 13 2015-05-11-13:43:00            0             0
 14 2015-05-11-13:44:00         2000          2000
 15 2015-05-11-13:45:00            0             0
 16 2015-05-11-13:46:00            0             0
 17 2015-05-11-13:47:00         1000          1000
 18 2015-05-11-13:48:00         1000          1000
 19 2015-05-11-13:49:00            0             0
 20 2015-05-11-13:50:00            0             0
```

15. To run a **compareNodes** query for the *cpu_user* metric, issue the following command:

```
# mmperfmon query compareNodes cpu_user
```

A sample output is as follows:

```
Legend:
1:      cluster1.ibm.com|CPU|cpu_user
2:      cluster2.ibm.com|CPU|cpu_user

Row           Timestamp cluster1 cluster2
  1 2015-05-11-13:53:54      0.5     0.25
  2 2015-05-11-13:53:55      0.5     0.25
  3 2015-05-11-13:53:56     0.25     0.25
  4 2015-05-11-13:53:57      0.5     0.25
  5 2015-05-11-13:53:58     0.25     0.75
```

```
  6 2015-05-11-13:53:59      0.5     0.25
  7 2015-05-11-13:54:00     0.25     0.25
  8 2015-05-11-13:54:01      0.5     0.25
  9 2015-05-11-13:54:02     0.25     0.25
 10 2015-05-11-13:54:03      0.5     0.25
```

16. To run an object query, issue the following command:

```
# mmperfmon query objObj 2016-09-28-09:56:39 2016-09-28-09:56:43
```

A sample output is as follows:

```
1: cluster1.ibm.com|SwiftObject|object_auditor_time
2: cluster1.ibm.com|SwiftObject|object_expirer_time
3: cluster1.ibm.com|SwiftObject|object_replication_partition_delete_time
4: cluster1.ibm.com|SwiftObject|object_replication_partition_update_time
5: cluster1.ibm.com|SwiftObject|object_DEL_time
6: cluster1.ibm.com|SwiftObject|object_DEL_err_time
7: cluster1.ibm.com|SwiftObject|object_GET_time
8: cluster1.ibm.com|SwiftObject|object_GET_err_time
9: cluster1.ibm.com|SwiftObject|object_HEAD_time
10: cluster1.ibm.com|SwiftObject|object_HEAD_err_time
11: cluster1.ibm.com|SwiftObject|object_POST_time
12: cluster1.ibm.com|SwiftObject|object_POST_err_time
13: cluster1.ibm.com|SwiftObject|object_PUT_time
14: cluster1.ibm.com|SwiftObject|object_PUT_err_time
15: cluster1.ibm.com|SwiftObject|object_REPLICATE_time
16: cluster1.ibm.com|SwiftObject|object_REPLICATE_err_time
17: cluster1.ibm.com|SwiftObject|object_updater_time

Row object_auditor_time object_expirer_time object_replication_partition_delete_time
object_replication_partition_update_time object_DEL_time object_DEL_err_time object_GET_time
object_GET_err_time object_HEAD_time object_HEAD_err_time object_POST_time object_POST_err_time
object_PUT_time object_PUT_err_time object_REPLICATE_time object_REPLICATE_err_time object_updater_time
1 2016-09-28 09:56:39 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.855923
  0.000000 0.000000 0.000000 45.337915 0.000000 0.000000 0.000000 0.000000
2 2016-09-28 09:56:40 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
  0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
3 2016-09-28 09:56:41 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.931925
  0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
4 2016-09-28 09:56:42 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.855923
  0.000000 0.000000 0.000000 516.280890 0.000000 0.000000 0.000000 0.000000


object_DEL_total_time = 0.0          object_PUT_total_time = 561.618805
object_GET_total_time = 0.0          object_POST_total_time = 0.0
object_HEAD_total_time = 1.786948    object_PUT_max_time = 516.28089
object_POST_max_time = 0.0           object_GET_max_time = 0.0
object_HEAD_max_time = 0.931025      object_DEL_max_time = 0.0
object_GET_avg_time = 0.0            object_DEL_avg_time = 0.0
object_PUT_avg_time = 280.809402     object_POST_avg_time = 0.0
object_HEAD_avg_time = 0.893474      object_DEL_time_count = 0.0
object_POST_time_count = 0           object_PUT_time_count = 2
object_HEAD_time_count = 2           object_GET_time_count = 0
object_DEL_min_time = 0.0            object_PUT_min_time = 45.337915
object_GET_min_time = 0.0            object_POST_min_time = 0.0
object_HEAD_min_time = 0.855923
```

17. To view expired keys, issue the following command:

```
# mmperfmon query –list=expiredKeys
```

A sample output is as follows:

```


Found expired keys:
test_nodename|GPFSFilesystem|gpfsgui-cluster-2.novalocal|fs1
test_nodename|GPFSFilesystem|gpfsgui-cluster-2.novalocal|fs2
test_nodename|GPFSFilesystemAPI|gpfsgui-cluster-2.novalocal|fs2
test_nodename|GPFSFilesystemAPI|gpfsgui-cluster-2.novalocal|fs1
test_nodename|GPFSFilesystem|gpfsgui-cluster-2.novalocal|gpfs0
test_nodename|DiskFree|/mnt/gpfs0
test_nodename|Netstat
test_nodename|GPFSFilesystem|gpfsgui-cluster-2.novalocal|objfs
test_nodename|GPFSVFS
test_nodename|GPFSNode
test_nodename|GPFSFilesystemAPI|gpfsgui-cluster-2.novalocal|gpfs0
test_nodename|GPFSFilesystemAPI|gpfsgui-cluster-2.novalocal|objfs
test_nodename|DiskFree|/gpfs/fs2
test_nodename|DiskFree|/gpfs/fs1
test_nodename|GPFSRPCS
test_nodename|CPU
test_nodename|GPFSNodeAPI
test_nodename|Load
test_nodename|DiskFree|/mnt/objfs
test_nodename|Memory
```

18. To delete expired key, issue the following command:

```
# mmperfmon delete –key ‘test_nodename|DiskFree|/mnt/gpfs0‘
```

A sample output is as follows:

```
Check expired keys completed. Successfully 3 keys deleted.
```

19. To list a CPU report, issue the following command:

```
# mmperfmon report top --bucket-size 60 --number-buckets 1
```

A sample output is as follows:

```
[root@node-11 ~]# mmperfmon report top --bucket-size 60 --number-buckets 1
Top values in format:
{process name | PID} {cpu per mil} {memory per mil}

Time                 node-11.localnet.com
2020-04-03-17:11:00  11684 33 12
                     9205 23 302
                     26331 9 0
                     27671 9 1
                     1 3 2
                     9 3 0
                     386 1 0
                     707 1 0
                     1012 1 5
                     1016 1 2
[root@node-11 ~]#
```

See also

- mmdumpperfdata command

Location

/usr/lpp/mmfs/bin

mmpmon command

Manages performance monitoring and displays performance information.

Synopsis

```
mmpmon [-i CommandFile] [-d IntegerDelayValue] [-p]
  [-r IntegerRepeatValue] [-s] [-t IntegerTimeoutValue]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Before you attempt to use `mmpmon`, it is a good idea to review the current command topic and read the topic *Monitoring I/O performance with the mmpmon command* in the *IBM Spectrum Scale: Problem Determination Guide*.

Use the `mmpmon` command to manage GPFS performance monitoring functions and display performance monitoring data. The `mmpmon` command reads requests from an input file or standard input (stdin), and writes responses to standard output (stdout). Error messages go to standard error (stderr). Prompts, if not suppressed, go to stderr.

You can run `mmpmon` so that it continually reads input from a pipe. That is, the driving script or application never sends an end of file. In this scenario, it is a good idea to set the `-r` option to 1, or to use the default value of 1. This setting prevents the command from caching input records. In doing so it avoids unnecessary memory consumption.

This command cannot be run from a Windows node.

Results

The performance monitoring request is sent to the GPFS daemon that is running on the same node that is running the `mmpmon` command.

All results from the request are written to stdout.

The command has two output formats:

- Human readable, intended for direct viewing.

  In this format, the results are keywords that describe the value presented, followed by the value. Here is an example:

  ```
  disks: 2
  ```

- Machine readable, an easily parsed format intended for further analysis by scripts or applications.

  In this format, the results are strings with values presented as keyword/value pairs. The keywords are delimited by underscores (_) and blanks to make them easier to locate.

For details on how to interpret the `mmpmon` command results, see the topic *Monitoring GPFS I/O performance with the mmpmon command* in the *IBM Spectrum Scale: Administration Guide*.

Parameters

-i *CommandFile*
: The input file contains `mmpmon` command requests, one per line. Use of the `-i` flag implies use of the `-s` flag. For interactive use, omit the `-i` flag. In this case, the input is then read from stdin, allowing `mmpmon` to take keyboard input or output piped from a user script or application program.

Leading blanks in the input file are ignored. A line beginning with a number sign (#) is treated as a comment. Leading blanks in a line whose first non-blank character is a number sign (#) are ignored.

The input requests to the mmpmon command are as follows:

fs_io_s
Displays I/O statistics per mounted file system.

io_s
Displays I/O statistics for the entire node.

nlist add *name* [*name*...]
Adds node names to a list of nodes for mmpmon processing.

nlist del
Deletes a node list.

nlist new *name* [*name*...]
Creates a node list.

nlist s
Shows the contents of the current node list.

nlist sub *name* [*name*...]
Deletes node names from a list of nodes for mmpmon processing.

once *request*
Indicates that the request is to be performed only once.

qosio
Displays statistics for Quality of Service for I/O operations (QoS).

reset
Resets statistics to zero.

rhist nr
Changes the request histogram facility request size and latency ranges.

rhist off
Disables the request histogram facility. This value is the default.

rhist on
Enables the request histogram facility.

rhist p
Displays the request histogram facility pattern.

rhist reset
Resets the request histogram facility data to zero.

rhist s
Displays the request histogram facility statistics values.

rpc_s
Displays the aggregation of execution time for remote procedure calls (RPCs).

rpc_s size
Displays the RPC execution time according to the size of messages.

ver
Displays mmpmon version.

vio_s [f rg *RecoveryGroupName* [da *DeclusteredArray* [v *Vdisk*]]] [reset]
Displays IBM Spectrum Scale RAID vdisk I/O statistics. For more information about IBM Spectrum Scale RAID, see *IBM Spectrum Scale RAID: Administration*.

vio_s_reset [f rg *RecoveryGroupName* [da *DeclusteredArray* [v *Vdisk*]]]
Resets IBM Spectrum Scale RAID vdisk I/O statistics. For more information about IBM Spectrum Scale RAID, see *IBM Spectrum Scale RAID: Administration*.

loc_io_s [f fs *fsName* [p *poolName*]]
Displays locality I/O statistics. It displays the amount of data that is written/read locally and remotely. It supports file system filter and pool filter.

Options

-d *IntegerDelayValue*
Specifies a number of milliseconds to sleep after one invocation of all the requests in the input file. The default value is 1000. This value must be an integer greater than or equal to 500 and less than or equal to 8000000.

The command processes the input file in the following way:

1. The command processes each request in the input file sequentially. It reads a request, processes it, sends it to the GPFS daemon, and waits for the response. When it receives the response, the command processes it and displays the results of the request. The command then goes on to the next request in the input file.
2. When the command processes all the requests in the input file, it sleeps for the specified number of milliseconds.
3. When the command wakes, it begins another cycle of processing, beginning with Step 1. The number of repetitions depends on the value of the `-r` flag.

-p
Indicates to generate output that can be parsed by a script or program. If this option is not specified, human-readable output is produced.

-r *IntegerRepeatValue*
Specifies the number of times to run all the requests in the input file.

The default value is one. Specify an integer between zero and 8000000. Zero means to run forever, in which case processing continues until it is interrupted. This feature is used, for example, by a driving script or application program that repeatedly reads the result from a pipe.

The `once` prefix directive can be used to override the `-r` flag. See the description of `once` in the topic *Monitoring GPFS I/O performance with the mmpmon command* in the *IBM Spectrum Scale: Administration Guide*.

-s
Indicates to suppress the prompt on input.

Use of the `-i` flag implies use of the `-s` flag. For use in a pipe or with redirected input (<), the `-s` flag is preferred. If not suppressed, the prompts go to standard error (stderr).

-t *IntegerTimeoutValue*
Specifies a number of seconds that the command waits for responses from the GPFS daemon before it fails.

The default value is 60. This value must be an integer greater than or equal to 1 and less than or equal to 8000000.

Exit status

0
Successful completion.

1
Various errors, including insufficient memory, input file not found, incorrect option, and others.

3
Either no commands were entered interactively, or the input file did not contain any `mmpmon` commands. The input file was empty, or consisted of all blanks or comments.

4
`mmpmon` terminated due to a request that was not valid.

5
An internal error occurred.

111
An internal error occurred. A message follows.

Restrictions

1. Up to five instances of mmpmon can be run on a node concurrently. However, concurrent users might interfere with each other. See the topic *Monitoring GPFS I/O performance with the mmpmon command* in the *IBM Spectrum Scale: Administration Guide*.
2. Do not alter the input file while mmpmon is running.
3. The input file must contain valid input requests, one per line. When mmpmon finds an invalid request, it issues an error message and terminates. The command processes input requests that appear in the input file before the first invalid request.

Security

The mmpmon command must be run by a user with root authority, on the node for which you want statistics.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. Assume that `infile` contains these requests:

```
ver
io_s
fs_io_s
rhist off
```

   The following command is issued:

```
mmpmon -i infile -r 10 -d 5000
```

   The output (sent to stdout) is similar to the following example output:

```
mmpmon node 192.168.1.8 name node1 version 3.1.0
mmpmon node 192.168.1.8 name node1 io_s OK
timestamp:      1083350358/935524
bytes read:              0
bytes written:           0
opens:                   0
closes:                  0
reads:                   0
writes:                  0
readdir:                 0
inode updates:           0
mmpmon node 192.168.1.8 name node1 fs_io_s status 1
no file systems mounted
mmpmon node 192.168.1.8 name node1 rhist off OK
```

   The requests in the input file are run 10 times, with a delay of 5000 milliseconds (5 seconds) between invocations.

2. This example uses the same parameters as the previous example, but with the `-p` flag:

```
mmpmon -i infile -p -r 10 -d 5000
```

   The output (sent to stdout) is similar to the following example output:

```
_ver_ _n_ 192.168.1.8 _nn_ node1 _v_ 2 _lv_ 3 _vt_ 0
_io_s_ _n_ 192.168.1.8 _nn_ node1 _rc_ 0 _t_ 1084195701 _tu_ 350714 _br_ 0 _bw_ 0 _oc_ 0
     _cc_ 0 _rdc_ 0 _wc_ 0 _dir_ 0 _iu_ 0
_fs_io_s_ _n_ 192.168.1.8 _nn_ node1 _rc_ 1 _t_ 1084195701 _tu_ 364489 _cl_ - _fs_ -_rhist_
     _n_ 192.168.1.8 _nn_ node1 _req_ off _rc_ 0 _t_ 1084195701 _tu_ 378217
```

3. This example uses the `fs_io_s` option with a mounted file system:

```
mmpmon node 198.168.1.8 name node1 fs_io_s OK
cluster: node1.localdomain
filesystem: gpfs1
disks: 1
timestamp: 1093352136/799285
bytes read: 52428800
bytes written: 87031808
opens: 6
closes: 4
reads: 51
writes: 83
readdir: 0
inode updates: 11

mmpmon node 198.168.1.8 name node1 fs_io_s OK
cluster: node1.localdomain
filesystem: gpfs2
disks: 2
timestamp: 1093352136/799285
bytes read: 87031808
bytes written: 52428800
opens: 4
closes: 3
reads: 12834
writes: 50
readdir: 0
inode updates: 9
```

4. This example is the same as the previous one, but with the `-p` flag:

```
_fs_io_s_ _n_ 198.168.1.8 _nn_ node1 _rc_ 0 _t_ 1093352061 _tu_ 93867 _cl_ node1.localdomain
  _fs_ gpfs1 _d_ 1 _br_ 52428800 _bw_ 87031808 _oc_ 6 _cc_ 4 _rdc_ 51 _wc_ 83 _dir_ 0 _iu_ 10
_fs_io_s_ _n_ 198.168.1.8 _nn_ node1 _rc_ 0 _t_ 1093352061 _tu_ 93867 _cl_ node1.localdomain
  _fs_ gpfs2 _d_ 2 _br_ 87031808 _bw_ 52428800 _oc_ 4 _cc_ 3 _rdc_ 12834 _wc_ 50 _dir_ 0
_iu_ 8
```

This output consists of two strings.

5. This example uses the `io_s` option with a mounted file system:

```
mmpmon node 198.168.1.8 name node1 io_s OK
timestamp: 1093351951/587570
bytes read: 139460608
bytes written: 139460608
opens: 10
closes: 7
reads: 12885
writes: 133
readdir: 0
inode updates: 14
```

6. This example is the same as the previous one, but with the `-p` flag:

```
_io_s_ _n_ 198.168.1.8 _nn_ node1 _rc_ 0 _t_ 1093351982 _tu_ 356420 _br_ 139460608
  _bw_ 139460608 _oc_ 10 _cc_ 7 _rdc_ 0 _wc_ 133 _dir_ 0 _iu_ 14
```

This output consists of one string.

7. This example uses an `loc_io_s` option to display the total size of local read data without filter.

```
mmpmon node 192.168.116.106 name localhost loc_io_s OK
cluster:                        sncfs.gpfs.net
filesystem:                             *
pool:                                   *
timestamp:                      1494238949/181786
bytes read:                       24158208
local bytes read:                  9236480
bytes written:                       65536
local bytes written:                 32768
reads:                                1914
local reads:                           714
writes:                                 16
local writes:                            8
```

8. This example is same as the previous one, but with `f fs sncfs p datapool` filters.

```
mmpmon node 192.168.116.106 name localhost loc_io_s OK
cluster:                        sncfs.gpfs.net
filesystem:                          sncfs
pool:                              datapool
timestamp:                      1494238956/323617
bytes read:                         114688
local bytes read:                    59392
bytes written:                           0
local bytes written:                     0
reads:                                  27
local reads:                            14
writes:                                  0
local writes:                            0
```

Location

/usr/lpp/mmfs/bin

mmprotocoltrace command

Starts, stops, and monitors tracing for the CES protocols.

Synopsis

```
mmprotocoltrace start Identifier [Identifier...] [-c ClientIP]
                     [-d Duration] [-l LogFileDir] [-N Nodes] [-f]
```

or

```
mmprotocoltrace stop [Identifier [Identifier...]]
```

or

```
mmprotocoltrace status [Identifier [Identifier...]] [-v]
```

or

```
mmprotocoltrace clear [Identifier [Identifier...]] [-f]
```

or

```
mmprotocoltrace reset Identifier [Identifier...]
```

or

```
mmprotocoltrace {config}
```

Availability

Available on all IBM Spectrum Scale editions.

Notice: This command has common function to other existing commands. As such, the function may, at any time in a future release, be rolled into other commands and immediately deprecated from use without prior notice. Information about the change and what commands replace it would be provided in some format at the time of that change. Users should avoid using this command in any type of automation or scripting or be advised a future change may break that automation without prior notice.

Description

Use the `mmprotocoltrace` command to trace Winbind, SMB, or network operations. You can start, stop, reset, or display the status of a trace with this command. It also controls the timeouts for the traces to avoid excessive logging.

Note: The protocol functions provided in this command, or any similar command, are generally referred to as CES (Cluster Export Services). For example, protocol node and CES node are functionally equivalent terms.

For more information about this command, see the topic *CES tracing and debug data collection* in the *IBM Spectrum Scale: Problem Determination Guide*.

Parameters

options

Specifies one of the following trace options:

-d *Duration*

Specifies the trace duration in minutes. The default is 10.

-l *LogFileDir*
Specifies the name of the directory that contains the log and tar files that are created by the trace. The directory name cannot be a shared directory. The default is a directory in `/tmp/mmfs` that is named by the trace type and time.

If the sudo wrapper mode is used, then the internal value for `LogFileDir` is set to the value that is configured for `dataStructureDump` in the IBM Spectrum Scale configuration using the **`mmlsconfig dataStructureDump`** command.

-N *Nodes*
Specifies a comma-separated list of names of the CES nodes where you want tracing to be done. The default is all the CES nodes. For more information, see the topic *Tips for using mmprotocoltrace* in the *IBM Spectrum Scale: Problem Determination Guide*.

-c *ClientIPs*
Specifies a comma-separated list of client IP addresses to trace. The CES nodes that you specified in the -N parameter will trace their connections with these clients. This parameter applies only to SMB traces and Network traces. For more information, see the topic *Tips for using mmprotocoltrace* in the *IBM Spectrum Scale: Problem Determination Guide*.

-f
Forces an action to occur. Affects the **`clear`** command. This parameter also disables prompt for smb and smbsyscalls.

-v
Verbose output. Affects only the **`status`** command.

command
Specifies one of the following trace commands:

start
Starts tracing for the specified component.

stop
Stops tracing for the specified component. If no component is specified, this option stops all active traces.

status
Displays the status of the specified component. If no component is specified, this option displays the status of all current traces.

config
Displays the protocol tracing configuration settings that are currently active for the node on which the command is run. For example, the limit for the maximal trace size. These settings are defined in the file `/var/mmfs/ces/mmprotocoltrace.conf` and are specific for each CES node.

clear
Clears the trace records from the trace list. If no component is specified, this option clears all current traces.

reset
Resets the nodes to the default state that is defined in the configuration file.

Identifier
Specifies one of the following components:

smb
Traces the SMB service. This enables a detailed tracing for incoming SMB connections from the specified SMB client IP addresses. When using this trace, it is recommended to only have few incoming connections from the specified IP addresses, as too many traced connections can impact the cluster performance.

network
Traces the Network service.

smbsyscalls
Collects the strace-logs for SMB. This collects a trace for all system calls issued from SMB connections from the specified SMB client IP addresses. When using this trace it is recommended

to only have few incoming connections from the specified IP addresses, as too many traced connections can impact the cluster performance.

winbind
Traces the winbind service that is used for user authentication.

Exit status

0
Successful completion.

Nonzero
A failure occurred.

Security

You must have root authority to run the `mmprotocoltrace` command.

The node on which the command is run must be able to process remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. See the information about the requirements for administering a GPFS system in the *IBM Spectrum Scale: Administration Guide*.

Examples

1. To start an SMB trace, issue the following command:

```
[root@ces5040-41 ~]# mmprotocoltrace start smb -c 10.0.100.42,10.0.100.43
```

A sample output is as follows:

```
Starting traces for the given clients will stop their connections prior to tracing.
Open files on these connections might get corrupted so please close them first.
Do you really want to perform the operation? (yes/no - default no): yes
Setting up traces
Trace 'f5d75a67-621e-4f09-8d00-3f9efc4093f2' created successfully for 'smb'

Waiting for all participating nodes...

Trace ID:           f5d75a67-621e-4f09-8d00-3f9efc4093f2
State:              ACTIVE
Protocol:           smb
Start Time:         15:24:30 25/09/19
End Time:           15:34:30 25/09/19
Trace Location:     /tmp/mmfs/smb.20190925_152430.trc
Origin Node:        ces5040-41.localnet.com
Client IPs:         10.0.100.42, 10.0.100.43
Syscall:            False
Syscall Only:       False
Nodes:
      Node Name:            ces5040-41.localnet.com
      State:                ACTIVE

      Node Name:            ces5040-42.localnet.com
      State:                ACTIVE

      Node Name:            ces5040-43.localnet.com
      State:                ACTIVE
```

2. To stop the SMB trace, issue the following command:

```
[root@ces5040-41 ~]# mmprotocoltrace stop smb
```

A sample output is as follows:

```
Stopping traces

Trace 'f5d75a67-621e-4f09-8d00-3f9efc4093f2' stopped for smb

Waiting for all participating nodes...
```

```
Collecting data from the participating nodes 'f5d75a67-621e-4f09-8d00-3f9efc4093f2'
Collected 'ces5040-43.localnet.com:/tmp/mmfs/smb.20190925_152430.trc'
Collected 'ces5040-42.localnet.com:/tmp/mmfs/smb.20190925_152430.trc'
Collected 'ces5040-41.localnet.com:/tmp/mmfs/smb.20190925_152430.trc'
Collected 'ces5040-44.localnet.com:/tmp/mmfs/smb.20190925_152430.trc'
Trace tar file has been written to '/tmp/mmfs/smb.trace.20190925_152552.tar.gz'
```

See also

- “mmaddcallback command” on page 12
- “mmces command” on page 133
- “mmchconfig command” on page 170
- “mmlscluster command” on page 492
- “mmlsconfig command” on page 495
- “mmsmb command” on page 709
- “mmuserauth command” on page 738

Location

/usr/lpp/mmfs/bin

mmpsnap command

Creates or deletes identical snapshots on the cache and home clusters, or shows the status of snapshots that have been queued up on the gateway nodes.

Synopsis

```
mmpsnap Device create -j FilesetName [{[--comment Comment] [--uid ClusterUID]} | --rpo] [--wait]
```

or

```
mmpsnap Device delete -s SnapshotName -j FilesetName
```

or

```
mmpsnap Device status -j FilesetName
```

Availability

Available on all IBM Spectrum Scale editions. Available on AIX and Linux.

Description

The `mmpsnap` command creates or deletes identical snapshots on the cache and home clusters, or shows the status of snapshots that have been queued up on the gateway nodes. You can use this command only in a Single writer (SW) cache. You cannot use for Read only (RO), Independent writer (IW), or Local updates (LU) caches. Peer snapshots are not allowed on a Single writer (SW) cache that uses the NSD protocol for communicating with home.

Parameters

Device
Specifies the device name of the file system.

create
Creates a fileset level snapshot in cache and a snapshot with the same name at home. The snapshot at home could be fileset level or file system level, depending on whether the exported path is an independent fileset or file system.

-j *FilesetName*
Specifies the name of the fileset.

--comment *Comment*
Optional; specifies user-defined text to be prepended to the snapshot name (thereby customizing the name of the snapshot).

Note: You must use alphanumeric characters when you customize the snapshot name.

--uid *ClusterUID*
Optional; specifies a unique identifier for the cache site. If not specified, this defaults to the GPFS cluster ID.

--rpo
Optional; specifies that user recovery point objective (RPO) snapshots are to be created for a primary fileset. This option cannot be specified with the `--uid` option.

--wait
Optional; makes the creation of cache and home snapshots a synchronous process. When specified, `mmpsnap` does not return until the snapshot is created on the home cluster. When not specified, `mmpsnap` is asynchronous and returns immediately rather than waiting for the snapshot to be created at home.

delete
Deletes the local and remote copies of the specified snapshot; AFM automatically figures out the remote device and fileset.

-s *SnapshotName*
Specifies the name of the snapshot to be deleted. A snapshot name is constructed as follows:

```
{commentString}-psnap-{clusterId}-{fsUID}-{fsetID}-{timestamp}
```

Where *timestamp* has the form YY-MM-DD-HH-MM-SS.

In the following example, a comment string was not provided:

```
psnap-14133737607146558608-C0A8AA04:4EDD34DF-1-11-12-05-14-32-10
```

status
Shows the status of snapshots that have been queued up on the gateway nodes. The status includes the following pieces of information:

- Last successful snapshot (obtained through `mmlsfileset --afm`)
- Status of the current snapshot process.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmpsnap` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To create a fileset level snapshot in cache of a single-writer fileset called `sw` in file system `fs1` issue this command:

```
mmpsnap fs1 create -j sw
```

The system displays output similar to the following:

```
Writing dirty data to disk.
Quiescing all file system operations.
Writing dirty data to disk again.
Snapshot psnap-13882361812785649740-C0A80E85:4F44B305-59-12-03-01-02-27-28 created with id 8.
Snapshot psnap-13882361812785649740-C0A80E85:4F44B305-59-12-03-01-02-27-28 created at the
satellite.
 Core snapshot has been queued.
```

2. To display the snapshot issue this command:

```
mmlssnapshot fs1 -j sw
```

The system displays output similar to the following:

```
Snapshots in file system fs1:
Directory SnapId Status Created Fileset
```

```
psnap-13882361812785649740-C0A80E85:4F44B305-59-12-03-01-02-27-28 8 Valid Thu Mar 1 02:27:29
2012 sw
```

3. To show that the snapshot is also created at home, issue this command:

```
mmlssnapshot fs1
```

The system displays output similar to the following:

```
Snapshots in file system fs1:
Directory SnapId Status Created Fileset
psnap-13882361812785649740-C0A80E85:4F44B305-59-12-03-01-02-27-28 8 Valid Thu Mar 1 02:23:16
2012
```

See also

- “mmafmconfig command” on page 45
- “mmafmctl command” on page 61
- “mmafmlocal command” on page 78
- “mmchattr command” on page 157
- “mmchconfig command” on page 170
- “mmchfileset command” on page 224
- “mmchfs command” on page 232
- “mmcrfileset command” on page 311
- “mmcrfs command” on page 318
- “mmlsconfig command” on page 495
- “mmlsfileset command” on page 501
- “mmlsfs command” on page 506

Location

`/usr/lpp/mmfs/bin`

mmputacl command

Sets the GPFS access control list for the specified file or directory.

Synopsis

```
mmputacl [-d] [-i InFilename] Filename
```

Availability

Available on all IBM Spectrum Scale editions. Available on AIX and Linux.

Description

Use the `mmputacl` command to set the ACL of a file or directory.

If the `-i` option is not used, the command expects the input to be supplied through standard input, and waits for your response to the prompt.

For information about NFS V4 ACLs, see the topic *Managing GPFS access control lists* in the *IBM Spectrum Scale: Administration Guide*.

Any output from the `mmgetacl` command can be used as input to `mmputacl`. The command is extended to support NFS V4 ACLs. In the case of NFS V4 ACLs, there is no concept of a default ACL. Instead, there is a single ACL and the individual access control entries can be flagged as being inherited (either by files, directories, both, or neither). Consequently, specifying the `-d` flag for an NFS V4 ACL is an error. By its nature, storing an NFS V4 ACL implies changing the inheritable entries (the GPFS default ACL) as well.

The following describes how `mmputacl` works for POSIX and NFS V4 ACLs:

```
Command       POSIX ACL                      NFS V4 ACL
-----------   ----------------------------   ------------------------
mmputacl      Access ACL (Error if default   Stores the ACL (implies
               ACL is NFS V4 [1])             default as well)
mmputacl -d   Default ACL (Error if access   Error: NFS V4 ACL (has
               ACL is NFS V4 [1]              no default ACL)
-------------------------------------------------------------------
[1] The default and access ACLs are not permitted to be mixed types
    because NFS V4 ACLs include inherited entries, which are the
    equivalent of a default ACL.  An mmdelacl of the NFS V4 ACL is
    required before an ACL is converted back to POSIX.
-------------------------------------------------------------------
```

Depending on the file system's `-k` setting (`posix`, `nfs4`, or `all`), `mmputacl` may be restricted. The `mmputacl` command is not allowed to store an NFS V4 ACL if `-k posix` is in effect. The `mmputacl` command is not allowed to store a POSIX ACL if `-k nfs4` is in effect. For more information, see the description of the `-k` flag for the `mmchfs`, `mmcrfs`, and `mmlsfs` commands.

Note that the test to see if the given ACL is acceptable based on the file system's `-k` setting cannot be done until after the ACL is provided. For example, if `mmputacl file1` is issued (no `-i` flag specified) the user then has to input the ACL before the command can verify that it is an appropriate ACL given the file system settings. Likewise, the command `mmputacl -d dir1` (again the ACL was not given with the `-i` flag) requires that the ACL be entered before file system ACL settings can be tested. In this situation, the `-i` flag may be preferable to manually entering a long ACL, only to find out it is not allowed by the file system.

Parameters

Filename
The path name of the file or directory for which the ACL is to be set. If the `-d` option is specified, *Filename* must be the name of a directory.

Options

-d
Specifies that the default ACL of a directory is to be set. This flag cannot be used on an NFS V4 ACL.

-i *InFilename*
The path name of a source file from which the ACL is to be read.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You may issue the `mmputacl` command only from a node in the GPFS cluster where the file system is mounted.

You must be the file or directory owner, the root user, or someone with control permission in the ACL, to run the `mmputacl` command.

Examples

To use the entries in a file named `standard.acl` to set the ACL for a file named `project2.history`, issue this command:

```
mmputacl -i standard.acl project2.history
```

where `standard.acl` contains:

```
user::rwxc
group::rwx-
other::--x-
mask::rw-c
user:alpha:rwxc
group:audit:rwx-
group:system:-w--
```

To confirm the change, issue this command:

```
mmgetacl project.history
```

The system displays information similar to:

```
#owner:paul
#group:design
user::rwxc
group::rwx-
other::--x-
mask::rw-c
user:alpha:rwxc
group:audit:rwx-
group:system:-w--
```

See also

- “mmeditacl command” on page 401
- “mmdelacl command” on page 360
- “mmgetacl command” on page 428

Location

`/usr/lpp/mmfs/bin`

mmqos command

Controls the Quality of Service for I/O operations (QoS) settings for a file system.

Synopsis

```
mmqos class create Device --class ClassName [--fileset FilesetName[,FilesetName...]]
```

or

```
mmqos class update Device --class ClassName [{ {add | replace} --fileset FilesetName[,FilesetName...] |
   --remove --fileset FilesetName}]
```

or

```
mmqos class delete Device --class ClassName
```

or

```
mmqos class set Device --class ClassName Attribute=Value[,Attribute=Value...]
```

or

```
mmqos class list Device [--config] [-Y]
```

or

```
mmqos throttle create Device --pool PoolName --class ClassName
   [{-N {Node | NodeClass} | -C {all | all_local | all_remote | ClusterName}}]
   [--maxiops MaxIOPS] [--maxmbs MaxMBS] [--force]
```

or

```
mmqos throttle update Device --pool PoolName --class ClassName
   [{-N {Node | NodeClass} | -C {all | all_local | all_remote | ClusterName}}]
   [--maxiops MaxIOPS] [--maxmbs MaxMBS] [--force]
```

or

```
mmqos throttle updateKey Device --pool PoolName --class ClassName
   [{-N {Node | NodeClass} | -C {all | all_local | all_remote | ClusterName}}]
   {-new-N {Node | NodeClass} | -new-C {all | all_local | all_remote | ClusterName} }
```

or

```
mmqos throttle delete Device --pool PoolName --class ClassName
   [{-N {Node | NodeClass} | -C {all | all_local | all_remote | ClusterName}}]
```

or

```
mmqos throttle list Device [{--pool PoolName | --class ClassName}] [-Y]
```

or

```
mmqos config set Device Attribute=Value[,Attribute=Value...]
```

or

```
mmqos config list Device [-Y]
```

or

```
mmqos filesystem init Device
```

or

```
mmqos filesystem enable Device
```

or

```
mmqos filesystem disable Device
```

or

```
mmqos filesystem reset Device
```

or

```
mmqos filesystem refresh Device
```

or

```
mmqos filesystem list [-Y]
```

or

```
mmqos filesystem status Device [-Y]
```

or

```
mmqos filesystem restore --restore-file RestoreFile
```

or

```
mmqos report list Device [--pool {all | PoolName}] [--seconds Seconds] [--sum-classes {yes | no}]
   [--sum-nodes {yes | no}] [{-Y | --fine-stats-display-range FineStatsDisplayRange}]
```

Availability

Available with IBM Spectrum Scale Advanced Edition, IBM Spectrum Scale Data Management Edition, IBM Spectrum Scale Developer Edition, or IBM Spectrum Scale Erasure Code Edition.

Description

With the `mmqos` command, you can regulate I/O accesses to disk storage pools to prevent I/O-intensive processes from dominating I/O access and significantly delaying other processes that access the pools. The disk storage pools can be the system storage pool or user storage pools.

- With the QoS system classes, you can prevent certain I/O-intensive long running IBM Spectrum Scale maintenance commands from dominating I/O access to storage pools. This feature allows the maintenance commands to be run during regular system operation without significantly slowing other processes that access the same storage pools.
- With QoS user classes, you can regulate I/O access to files in filesets. This feature allows you to assign different allotments of I/O access to groups in your business organization, such as departments or task forces, based on their business priorities. For example, you can put the frequently accessed files of Department HR into filesets FS01 and FS02. You can then assign FS01 and FS02 to a QoS user class CL01 and set an I/O limit per second for the class. At run time, the QoS component allows I/O accesses per second to filesets FS01 and FS02 up to the I/O limit per second that you set for class CL01.

You can assign I/O limits to QoS classes either as I/O operations per second (IOPS) or as megabytes per second (MB/s). You can display regular or fine-grained statistics of I/O accesses by processes over time.

The `mmqos` command provides all the features that were introduced with the `mmchqos` and `mmlsqos` commands, supports QoS user classes, and has an easy-to-use command syntax.

Warning: `mmqos` is the future command for all QoS functions and it was built to replace the configuration capability of the **`mmchqos`** command. You can only use either **`mmchqos`** or **`mmqos`** on a file system at any time, but not both. If you start to configure a QoS file system with **`mmchqos`** or

having been using it for a while and you try to use **mmqos** to change that configuration, **mmqos** will issue you a usage error. If you start to configure a QoS file system with the new **mmqos** command and you try to use **mmchqos** to change that configuration, **mmchqos** will issue you a usage error. If you are ready to switch to **mmqos** from **mmchqos**, use the following commands to remove the existing **mmchqos** configuration:

```
mmchqos Device --reset --stat-poll-interval 0 --stat-slot-time 0
mmchqos Device --disable
```

Both commands have reset functions to remove their configuration with a file system. After removing the existing configuration, you can move to using the other command.

For more information, see *Setting the Quality of Service for I/O operations (QoS)* in the *IBM Spectrum Scale: Administration Guide*.

QoS user classes

With the QoS user classes, you can regulate I/O access to the system pool or to user storage pools by functional groups in your business organization, such as departments, projects, or task forces. Follow these general steps for each functional group:

1. Put files that are frequently accessed by the group into one or more filesets.
2. Identify the disk storage pool where the contents of the files are stored.
3. Create a QoS user class and assign the filesets to it.
4. Create a QoS throttle for the user class. The throttle configuration includes two types of information:
   - The I/O context, which specifies the file system, the disk storage pool, the QoS user class including the filesets, and the nodes that members of the functional group will use to access the files in the fileset.
   - The I/O limits, which you can specify in MB/s or in IOPS.

When the throttle is created, QoS limits the collective IOPS or MB/s by all processes that access the files in the filesets to the I/O limits that are set in the throttle.

QoS system classes

The QoS system classes provide file-system-wide allocation of IOPS or MB/s to four broad categories of processes. The disk storage pool can be the system storage pool or a user storage pool.

`maintenance` class

The `maintenance` class consists of the following I/O-intensive long running IBM Spectrum Scale maintenance commands:

`mmadddisk`	`mmdeldisk`	`mmfileid`	`mmlssnapshot`
`mmapplypolicy`	`mmdelfileset`	`mmfsck`	`mmrestripefs`
`mmbackup`	`mmdelsnapshot`	`mmimgbackup`	`mmrpldisk`
`mmcheckquota`	`mmdf`	`mmimgrestore`	

Typically you assign a smaller share of IOPS or MB/s to the `maintenance` class than you do to the class named `other`. A smaller share of IOPS or MB/s prevents these maintenance commands from dominating file system performance and significantly delaying other processes that use I/O.

When the `maintenance` class is configured with I/O limits, QoS restricts the processes in the class from collectively consuming more I/O resources than are set in the I/O limits. The default limit is `unlimited`, which means that QoS does not restrict the I/O at all.

other class

The other class includes almost all other processes that use file system I/O. Typically you assign a larger share of IOPS or MB/s or the constant unlimited to this class so that normal processes have greater access to I/O resources and finish more quickly.

The mmchdisk command by default runs in the class named other. However, when you issue the mmchdisk command you can specify that you want to assign it to the maintenance class. This assignment is effective only for the instance of the mmchdisk command that you are starting. Use this option in situations in which you need to limit the I/O of the mmchdisk command so that normal processes that use I/O can finish more quickly. For more information, see "mmchdisk command" on page 212.

When the other class is configured with I/O limits, QoS restricts the processes in the class from collectively consuming more I/O resources than are set in the I/O limits. The default limit is unlimited, which means that QoS does not restrict the I/O at all.

misc class

The misc class stores the count of IOPS or MB/s that some critical file system processes consume. You cannot assign IOPS or MB/s to this class, but its count of IOPS or MB/s is displayed along with the statistics of the other classes by the mmqos report list command.

mdio-all-sharing class

The mdio-all-sharing class is used for class sharing. For more information, see the subtopic "Class sharing" later in this topic.

Note: Enabling and disabling *mmqos QoS* services is somewhat analogous to **mmshutdown** and **mmstartup** for the file system. The QoS services within the daemon are either started or stopped and the action to enable or disable then should be infrequent.

Each invocation of the **mmqos** command that changes the configuration with actions like create, update, delete, set, enable, or disable causes the daemon to reread the QoS configuration from the CCR. If the *mmqos QoS* services are enabled, it causes a small delay of a couple of seconds in throttling accuracy, as new statistics are generated with the updated configuration. If the *mmqos QoS* services are enabled, it might take additional time to process these actions. That is, large changes to the configuration is more efficient when the *mmqos QoS* services are disabled.

Creating and modifying throttles

A throttle is a specification for limiting I/O for a QoS class. Each throttle consists of two parts: an I/O context and an I/O limit. As the following table shows, the I/O context consists of a file system, a disk pool, a QoS class, and a set of nodes. These elements occur as options in the mmqos throttle commands:

Table 28. Elements of an I/O context

Option	Element	Valid values	Default value	Create and modify
Device	File system	An existing file system.	These are required elements of the I/O context. There are no default values.	• Create: These elements must be specified in the mmqos throttle create command. • Modify: These elements cannot be changed after the throttle is created. To make a change, you must delete the throttle and re-create it with different elements.
--pool	Disk pool	An existing disk pool.		
--class	A QoS user class or a QoS system class	An existing QoS user class that was created or updated with one or more filesets, which become part of the I/O context.		
		QoS system classes: • maintenance • other • mdio-all-sharing		

Table 28. Elements of an I/O context (continued)				
Option	**Element**	**Valid values**	**Default value**	**Create and modify**
`-N` or `-C` Either option can be used to identify the nodes that the throttle applies to.	`-N` specifies one or more nodes.	*Node* or *NodeClass* (which includes class `all`)	All local nodes.	• Create: `mmqos throttle create` command. • Modify: `mmqos throttle updatekey` command. Only one of `-N` or `-C` can be specified in an `mmqos` command. The one that is not specified is ignored.
	`-C` refers to all the nodes in the specified cluster or clusters.	`all, all_local, all_remote`, or *ClusterName*	All local and remote clusters.	

For more information, see the description of the `mmqos throttle create` command later in this topic.

Notes:

- Throttle scope
  - The throttle scope identifies the nodes across which the I/O limit of the throttle is to be applied. The throttle scope can be set by either the `-N` option or the `-C` option.
  - For all QoS user classes and the QoS system class `other`, the default initial throttle scope is all nodes, both local and remote. For the QoS system class `maintenance`, the default initial throttle scope is `all_local`. This setting means that the I/O limits are applied across all the nodes in the local cluster.
  - If the throttle scope is not specified in the `-N` option or the `-C` option of the `mmqos throttle create` command, then the default value is `all_local` for the `maintenance` class and `all` for all other classes. The setting `all_local` means all nodes in the local cluster that have mounted the file system. The setting `all` means all nodes in the local cluster and in remote clusters that have mounted the file system.
- Specifying the correct throttle description
  - In the `mmqos throttle update` and the `mmqos throttle delete` commands, be sure to specify the correct throttle description, including the file system, pool, class, and throttle scope. For example, suppose that you created the following throttle, which has an I/O limit of 100 MB/s:

    ```
    # mmqos throttle create qos2 --pool system --class HR_Dept2 -N TCTNodeClass1 --maxmbs 100
    ```

    If you later issue the `mmqos throttle update` command to change the I/O limit of this throttle to 200 MB/s, you must specify exactly the same throttle description:

    ```
    # mmqos throttle update qos2 --pool system --class HR_Dept2 -N TCTNodeClass1 --maxmbs 200
    ```

    If you changed the throttle scope to `-C all_local` after you created the throttle, you must specify the new throttle scope when you change the I/O limit:

    ```
    # mmqos throttle update qos2 --pool system --class HR_Dept2 -C all_local --maxmbs 200
    ```

    If you accepted the default throttle scope when you created the throttle and you never changed the throttle scope afterward, then you do not need to specify the `-N` option or the `-C` option in the `mmqos throttle update` command:

    ```
    # mmqos throttle create qos2 --pool system --class HR_Dept2 --maxmbs 100
    ...
    # mmqos throttle update qos2 --pool system --class HR_Dept2 --maxmbs 200
    ```
  - If you specify a non-existent throttle description, the command responds with an informative message.

– To see the current throttle descriptions, issue the `mmqos throttle list` command:

```
# mmqos throttle list qos2
 Pool Name      Class Name      Node/Nodeclass      Cluster Scope       MaxIOPS    MaxMBS
-------------------------------------------------------------------------------------------
 system         HR_Dept2        TCTNodeClass1       -                   unlimited  100
```

As the following table shows, the I/O limit of a throttle consists of an IOPS limit or an MB/s limit. These elements occur as options in the `mmqos throttle create` and `mmqos throttle update` commands:

Table 29. Elements of I/O limits

Option	Element	Valid values	Default value	Create or modify
`--maxiops`	Maximum I/O operations per second	0 - 1999999999 or `unlimited`	`unlimited`	`mmqos throttle create` `mmqos throttle update`
`--maxmbs`	Maximum megabytes per second of I/O	0 - 1999999999 or `unlimited`	`unlimited`	

Notes:

- The default value for either the maximum IOPS or the maximum MB/s of a throttle is `unlimited`. This setting indicates that QoS does not restrict the I/O of the nodes that are specified in the throttle. But to create a throttle, one of the 2 settings requires a non-default value to ensure throttling is implemented.
- Although the I/O limit can include both a maximum IOPS setting and a maximum MB/s setting, at run time the more restrictive setting controls the I/O usage. For example, if the I/O limit is unlimited IOPS and 40 MB/s, then at run time the nodes that are specified in the throttle are collectively limited to 40 MB/s of I/O.
- No advantage results from setting both the maximum IOPS and the maximum MB/s to specific values. If both types of limit are set, QoS tracks both the IOPS usage and the MB/s usage at run time and restricts the I/O when either the maximum IOPS setting or the maximum MB/s setting is reached. To avoid uncertainty about which limit might be controlling the I/O usage for the throttle, it is a good idea to specify either MB/s or IOPS but not both.
- The maximum IOPS or the maximum MB/s settings are divided among the nodes that are specified in the throttle. For more information, see the subtopic "QoS operation" later in this topic.

QoS operation

The following text describes how QoS regulates the consumption of a specified number of IOPS or MB/s by processes that access files in the disk storage pools of a QoS class. Where the term "IOPS" appears, the intended meaning is "either IOPS or MB/s":

- For each class, QoS divides the specified IOPS among the nodes in the throttle scope that have mounted the file system. If the allocation is static, QoS assigns an equal number of IOPS to each node and does not change the assignments. If the allocation is Mount Dynamic I/O, QoS periodically adjusts the assignment of IOPS to each node based on the relative frequency of I/O accesses to the disk storage pools of the class by the processes that are running on each node. For more information, see the topic "Mount dynamic I/O (MDIO)."
- If the class is a QoS user class, QoS regulates I/O accesses to files for which the following conditions are true:
  – The file is in a fileset that belongs to the specified user class.
  – The file data is located in the disk pool that is specified by the throttle.

  If the class is one of the QoS system classes `maintenance` or `other`, QoS regulates I/O accesses to files for which the following conditions are true:

  - The process that initiates the I/O access belongs to the specified QoS system class.
  - The file data is located in the disk pool that is specified by the throttle.

- A QoS component on each node monitors the I/O accesses of the processes that are running on the node. During each one-second period, for each QoS class, the QoS component allows a QoS I/O access if the IOPS or MB/s that are allocated to the node have not been exhausted during the current one-second period. Otherwise, QoS queues the I/O access until the next one-second period begins.
- When you change IOPS allocations, a brief delay due to reconfiguration occurs before QoS begins applying the new allocations.
- The following QoS operations apply to unmounting and mounting a file system:
  - QoS stops applying IOPS allocations to a file system when you unmount it and resumes when you remount it.
  - IOPS allocations persist across unmounting and remounting a file system.
  - When you mount a file system, a brief delay due to reconfiguration occurs before QoS starts applying allocations.

Mount dynamic I/O (MDIO)

QoS can dynamically balance the allocation of IOPS or MB/s among the nodes that have mounted the file system based on the relative proportions of attempted I/O accesses to the storage pool of the class by the processes running on each node. This feature is called MDIO. By default, MDIO is enabled for QoS user classes and is disabled for QoS system classes. You can enable or disable MDIO for a class with the `mmqos class set` command.

When MDIO is disabled for a class, QoS allocates the specified IOPS or MB/s of the class equally among all the nodes that have mounted the file system. QoS does not take into account the frequency of accesses to the storage pool of the class by the processes that are running on a node.

For example, suppose that MDIO is disabled in class CL01 and that the class is assigned 300 IOPS for storage pool P01. If three nodes A, B, and C have mounted the file system, QoS allocates 100 IOPS to each node for pool P01 and never changes the allocation. If the processes that are running on nodes B and C currently require only 50 IOPS of access to P01, their combined surplus of 100 IOPS goes unused. If node A requires a total of 200 IOPS of access to P01, it cannot draw the 100 IOPS that it needs from the unused IOPS on nodes B and C.

When MDIO is enabled for a class, QoS dynamically balances the allocation of IOPS or MB/s among the nodes that have mounted the file system. QoS determines the allocations based on the total number of attempted accesses to the storage pool of the class by processes on each node during an internally set time period. At the end of each time period, QoS calculates for each node the proportion of its attempted accesses to the total number of attempted accesses during the time period. If the proportions have changed since the end of the previous time period, QoS rebalances the allocations according to the new proportions.

For example, suppose that MDIO is now enabled in class C01 and that the class is again assigned 300 IOPS for storage pool P01. Again QoS initially assigns 100 IOPS to each of the nodes A, B, and C. However, QoS then monitors the number of attempted I/O accesses to pool P01 for class CL01 by the processes on each node. Suppose that during the most recent measurement period the number of attempted I/O accesses to P01 for CL01 were 240 from node A, 92 from node B, and 68 from node C, for a total of 400 attempted accesses. The proportions of the number of attempted accesses from each node compared to the total number of attempted accesses are .60 (240/400), .23 (92/400), and .17 (68/400). If these proportions are different from those of the previous time period, QoS rebalances the allocation of IOPS according to the new proportions. In this example 300 IOPS are specified for the class, so QoS allocates 180 (.60 x 300) IOPS to node A, 69 IOPS (.23 x 300) to node B, and 51 IOPS (.17 x 300) to node C.

Class sharing

Class sharing is an easy and powerful way to integrate throttles that use the same storage pool and to give them the ability to share surplus I/O accesses dynamically among themselves. To participate in class sharing, a class must be mdio-enabled. For more information, see the subtopic "Mount dynamic I/O (MDIO)".

The following steps provide an overview of class sharing:

1. Two or more existing throttles regulate I/O accesses to the same storage pool P01. You want to change the limit on their combined I/O accesses to P01 but keep the relative proportion of their I/O limits the same. You also want any unused I/O accesses to be shared with throttles that need more I/O accesses.

2. You can create a throttle for the QoS system class `mdio-all-sharing` that regulates I/O accesses to the common storage pool. Set the I/O limit of the throttle to the total number of IOPS or MB/s that you want to allow for the combined throttles.

3. Now the individual throttles are collectively limited to the I/O limit that you configured for the `mdio-all-sharing` throttle. Each individual throttle now has an actual I/O limit that is based on the ratio of its original I/O limit to the total original I/O limits of all the individual throttles. If one individual throttle has surplus I/O accesses, they are shared proportionally among the other individual throttles.

For example, suppose that you have three existing throttles that regulate I/O accesses to the system pool and that have I/O limits of 200 IOPS, 300 IOPS, and 500 IOPS for a total of 1000 IOPS. You create a throttle for the `mdio-all-sharing` class and the system pool and assign an I/O limit of 1200 IOPS to the new throttle. Now the three individual throttles are collectively limited to 1200 IOPS. The individual throttles now have actual I/O limits of 240 IOPS (200/1000 x 1200), 360 IOPS (300/1000 x 1200), and 600 IOPS (500/1000 x 1200). If at some point in time the third throttle has a surplus of 200 IOPS, then 80 IOPS of the surplus (200/500 x 200) are dynamically assigned to the first throttle and 120 IOPS of the surplus (300/500 x 200) are dynamically assigned to the second throttle.

QoS statistics

On each node, a QoS component collects I/O statistics for each process that accesses a QoS disk storage pool. Periodically, the QoS component sends a report of the accumulated statistics to a QoS manager node. A QoS manager node stores and manages the statistics of QoS classes and nodes so that the statistics can be displayed by an `mmqos report list` command.

As the number of statistics reports increases, it can become so great that the QoS manager node is unable to process reports in a timely manner. If so, the statistics that are reported by the `mmqos report list` command might not reflect the actual number of I/O accesses.

For each class, two QoS attributes affect the frequency with which reports are sent to a QoS manager node. These attributes are set by the `mmqos config` command:

`stat-poll-interval=`*Seconds*
: This attribute specifies the length of the time period during which class statistics are collected. The following events occur during this period:

  1. The QoS statistics variables for the class are reset to zero.
  2. Statistics are collected until the collection interval ends.
  3. The accumulated statistics are sent to a QoS manager node.

  The length of the collection period determines the fine-grainedness of data that is collected. The shorter that the collection period is, the more fine-grained the data becomes.

`stat-slot-time=`*Milliseconds*
: This attribute specifies the amount of time that QoS waits after the end of one `stat-poll-interval` before beginning the next. The purpose of this attribute is to reduce the number of statistics reports that are sent for the class to the QoS manager node. The longer that the slot time period is, the fewer are the class-statistic reports that are sent.

The number of statistics reports also increases or decreases as or more or fewer nodes mount the file system. To counter these effects, it is a good idea to adjust the values of the `stat-poll-interval` and the `stat-slot-time` attributes for each class as the number of nodes that mount the file system significantly increases or decreases. You can adjust the values manually for each class, or you can activate automatic adjustments for a class by setting both `stat-poll-interval` and `stat-slot-time` to zero. The following table shows the values that automatic adjustment sets. If you adjust the values manually, use the values in the table as guidelines:

Table 30. Automatic adjustments of `stat-poll-interval` and `stat-slot-time`

Number of nodes that have mounted the file system	**`stat-slot-time` (milliseconds)**	**`stat-poll-interval` (seconds)**
< 32	1000	5
< 64	2000	10
< 128	3000	15
< 256	4000	20
< 512	6000	30
< 1024	8000	40
< 2048	10000	50
< 4096	12000	60
< 8192	12000	60
< 16384	12000	60
16384 or more	24000	120

Limitations

The `mmqos` command is subject to the following limitations:

- The `mmqos` command is available only for the Linux operating system. However, The daemon supports QoS throttling functions on both Linux and AIX nodes within a cluster.
- The QoS system classes have the following limitations:
  - They cannot use filesets.
  - They do not support MDIO by default.
- The `mmqos filesystem reset` command does not make a backup copy of the QoS configuration before it deletes it.
- To avoid creating stale data within the **`mmqos`** configuration, you must first remove any QoS configuration information associated with or that references other IBM Spectrum Scale system objects, such as file system, pool, fileset, node, or cluster name. Ensure the following when you plan to remove these objects from the cluster:
  - Before you delete a fileset (**`mmdelfileset`**), issue the **`mmqos class delete`** command to remove the fileset from the **`mmqos`** configuration. If the class contains a fileset that is referenced by a throttle object, delete that throttle object first with the **`mmqos throttle delete`** command and then remove the class that contains the fileset.
  - Before you delete a node (**`mmdelnode`**), a storage pool (**`mmdeldisk`**), or a cluster name (**`mmremotecluster delete`**), issue the **`mmqos throttle delete`** command to remove the throttle object that contains any of those system objects from the QoS configuration.
  - Before you delete a file system, issue the **`mmqos filesystem reset`** command first to remove any QoS configuration information that is associated with the file system.

Note:

- No limit is imposed on the number of user classes, other than limits that might be imposed by the operating system or hardware.
- No limit is imposed on the number of filesets that can be associated with a user class, other than limits that might be imposed by the operating system or hardware.

Parameters

class
Commands for working with QoS classes.

Note: The `class create`, `class update`, `class updatekey`, and `class delete` commands apply only to QoS user classes, not to QoS system classes. You cannot create, update, do an update key on, or delete a QoS system class.

`create`
Creates a QoS user class.

Device
The device name of the file system.

`--class` ***ClassName***
The name of the user class.

[`--fileset` ***FilesetName*** **[,*FilesetName*...]]**
The names of zero or more filesets to be included in the user class. If you do not specify a fileset, you can add filesets to the class later with the `class update` add command.

update
Adds, replaces, or removes filesets from a QoS user class.

Device
The device name of the file system.

`--class` ***ClassName***
The name of the QoS user class to be updated.

{`--add` | `--replace` | `--remove`}
The action to be done.

`--add`
Adds one or more filesets to the class.

`--replace`
Removes all filesets from the class and adds the specified filesets to the class. If a class has ten filesets and you specify `--replace` with two filesets, the command removes the ten filesets and adds the two filesets that you specify.

`--remove`
Removes a fileset from the class. You can remove only one fileset at a time.

Note: If the class includes only one fileset, you cannot remove it. If you want to keep the class but delete the fileset, you can add a second fileset to the class and then remove the first fileset. If you do not want to keep the class, you can delete the class with the `class delete` command.

[`--fileset` ***FilesetName*** **[,*FilesetName*...]]**
The names of one or more filesets to be added to or removed from the QoS user class. If the command is `remove`, you can specify only one fileset per command invocation.

delete
Deletes a QoS user class. If a class has throttles, you must delete the throttles before you can delete the class.

Device
The device name of the file system.

--class ***ClassName***
The user class to be deleted.

set
Sets the value of an attribute for a QoS user class and system class.

Note: Other QoS attributes exist and are set at the file system level for all QoS user classes in the file system. For more information, see the description of the `mmqos config set` command.

Device
The device name of the file system.

--class ***ClassName***
The user class or system class for which the attributes are set.

Attribute=*Value*[, Attribute=*Value*...]
The following attribute can be set:

mdio-enabled={yes | no}
Enables or disables MDIO for the class. By default, the value of this class attribute shadows the value of the file-system-level `mdio-enabled` attribute. When you set the value of this class attribute here, the value becomes fixed and no longer shadows the value of the file-system level attribute.

By default, MDIO is enabled for QoS user classes and is disabled for QoS system classes.

For more information, see the subtopic "Mount dynamic I/O (MDIO)" earlier in this topic.

list
Lists the QoS system classes and QoS user classes that are defined in the specified file system. For user classes, the command also lists the filesets that belong to the class.

Device
The device name of the file system.

[--config]
Instead of listing the filesets for each user class, the command lists the configuration settings for each user class. For more information, see the description of the `mmqos class set` option.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

throttle
Commands for working with QoS throttles. These commands apply to both system classes and user classes. For more information, see the subtopic "Creating and modifying throttles" earlier in this topic.

create
Creates a QoS throttle for the specified file system, storage pool, and QoS class.

Device
The device name of the file system.

--pool [*PoolName* | default]
A storage pool.

--class ***ClassName***
A QoS user class or a QoS system class. For more information, see the subtopic "QoS operation" earlier in this topic.

-N {*Node* | *NodeClass* }
The nodes among which the IOPS or MB/s for this throttle are to be divided. Also referred to as the *throttle scope*. For more information, see the subtopic "Creating and modifying throttles" earlier in this topic.

Node
The specified node.

NodeClass
The nodes in the specified class. The node class `mount` is not supported.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

-C {all | all_local | all_remote | *ClusterName*}
The nodes among which the IOPS or MB/s for this throttle are to be divided. Also referred to as the *throttle scope*. For more information, see the subtopic "Creating and modifying throttles" earlier in this topic.

all
All nodes that have mounted the file system, no matter which cluster the node belongs to. This group includes nodes in the cluster that owns the file system and nodes that belong to clusters that requested access to the file system.

all_local
All nodes that have mounted the file system and that belong to the cluster that owns the file system.

all_remote
All nodes that have mounted the file system and that belong to any cluster that requested access to the file system.

ClusterName
All nodes that have mounted the file system and that belong to the specified cluster.

--maxiops *MaxIOPS*
The IOPS that are to be divided among the nodes that are governed by this throttle. The default value is `unlimited`. The valid range is 0 - `1999999999`. To specify a value less than 100, you must specify the `force` option. It is a good idea not to specify both an IOPS limit and an MB/s limit for the same throttle.

--maxmbs *maxMBS*
The megabytes per second of data transfer that are to be divided among the nodes that are governed by this throttle. The default value is `unlimited`. The valid range is 0 - `1999999999`. To specify a value less than 100, you must specify the `force` option. It is a good idea not to specify both an IOPS limit and an MB/s limit for the same throttle.

--force
Overrides the lower limit on the number of IOPS or MB/s that can be allocated to a throttle. QoS enforces a lower limit of 100 IOPS or 100 MB/s on the value that the `--maxiops` option or the `--maxmbs` option can be set to. Setting a value below the limit with the `--force` option typically causes the processes that are affected by the throttle to run for an indefinitely long time.

update
Updates the IOPS limit or the MB/s limit of the specified throttle.

Important: If you specified the `-N` or `-C` option in the `mmmqos throttle create` command or if you used the `-N` or `-C` option in the `mmqos throttle updatekey` command to change the throttle scope, you must specify the same `-N` or `-C` option here as the current throttle scope. If you did not specify the `-N` or the `-C` option in either command, you do not have to specify it here, because the default value is being used. For more information, see the subtopic "Creating and modifying throttles" earlier in this topic.

Device
The device name of the file system.

--pool [*PoolName* | default]
A storage pool. Be sure to specify the correct storage pool for the throttle that you want to update. For more information, see the subtopic "Creating and modifying throttles" earlier in this topic.

--class *ClassName*
A QoS user class or system class. Be sure to specify the correct class for the throttle that you want to update. For more information, see the subtopic "Creating and modifying throttles" earlier in this topic.

-N {*Node* | NodeClass }
The nodes among which the IOPS or MB/s for this throttle are to be divided. See the information about this option in the description of the `mmqos throttle create` command.

-C {all | all_local | all_remote | *ClusterName*}
The nodes among which the IOPS or MB/s for this throttle are to be divided. See the information about this option in the description of the `mmqos throttle create` command.

--maxiops *maxIOPS*
The IOPS limit for this throttle. See the information about this option in the description of the `mmqos throttle create` command.

--maxmbs *maxMBS*
The MB/s limit for this throttle. See the information about this option in the description of the `mmqos throttle create` command.

--force
Overrides the lower limit on the number of IOPS or MB/s that can be allocated to a throttle. See the information about this option in the description of the `mmqos throttle create` command.

Note:

- The four options such as `--pool` *PoolName*, `--class` *ClassName*, `[{ -N {Node | NodeClass} | -C {all | all_remote | ClusterName} }]` make up a unique key for throttle objects and this key cannot be updated. The throttle key must be deleted and can then be created again with new values.
- To change the attributes of the objects such as [`--maxIOPS` *MaxIOPS*] and [`--maxMBS` *MaxMBS*], provide the exact key values for which those values were set. Otherwise, you will get the following error:

```
mmqos: Processing the new QOS configuration...
mmqos: The QOS configuration record was not found. Nothing to update.
```

- If a key value is not set (because -N and -C are optional), it is still part of a key value but does not need to be supplied. That is, as these attributes are optional, if you did not specify them when creating the record, you should not specify them when updating it.

updatekey
Updates the throttle scope, which specifies the nodes across which the I/O limit of the throttle is to be applied. The throttle scope can be set by either the `-new-N` option or the `-new-C` option. For more information, see the subtopic "Creating and modifying throttles" earlier in this topic.

Important: If you specified the `-N` or `-C` option in the `mmmqos throttle create` command or if you used the `-N` or `-C` option in the `mmqos throttle updatekey` command to change the throttle scope, you must specify the same `-N` or `-C` option here as the current throttle scope. If you did not specify the `-N` or the `-C` option in either command, you do not have to specify it here, because the default value is being used. For more information, see the subtopic "Creating and modifying throttles" earlier in this topic.

Device
The device name of the file system.

--pool [*PoolName* | default]
A storage pool. Be sure to specify the correct storage pool for the throttle that you want to update. For more information, see the subtopic "Creating and modifying throttles" earlier in this topic.

--class [*ClassName* | default]
A QoS user class or system class. Be sure to specify the correct class for the throttle that you want to update. For more information, see the subtopic "Creating and modifying throttles" earlier in this topic.

-N {*Node* | NodeClass }
The current throttle scope. See the information about this option in the description of the `mmqos throttle create` command.

-C {all | all_local | all_remote | *ClusterName*}
The current throttle scope. See the information about this option in the description of the `mmqos throttle create` command.

-new-N {*Node* | NodeClass }
The new throttle scope. See the information about the `-N` option in the description of the `mmqos throttle create` command.

-new-C {all | all_local | all_remote | *ClusterName*}
The new throttle scope. See the information about the `-C` option in the description of the `mmqos throttle create` command.

delete
Deletes the specified throttle.

Important: If you specified the `-N` or `-C` option in the `mmmqos throttle create` command or if you used the `-N` or `-C` option in the `mmqos throttle updatekey` command to change the throttle scope, you must specify the same `-N` or `-C` option here as the current throttle scope. If you did not specify the `-N` or the `-C` option in either command, you do not have to specify it here, because the default value is being used. For more information, see the subtopic "Creating and modifying throttles" earlier in this topic.

Device
The device name of the file system.

--pool *PoolName*
The disk storage pool of the throttle that you want to delete. For more information, see the subtopic "Creating and modifying throttles" earlier in this topic.

--class *ClassName*
The QoS class for the throttle that you want to delete. For more information, see the subtopic "Creating and modifying throttles" earlier in this topic.

-N {*Node* | NodeClass }
The throttle scope of the throttle that you want to delete. See the information about this option in the description of the `mmqos throttle create` command.

-C {all | all_local | all_remote | *ClusterName*}
The throttle scope of the throttle that you want to delete. See the information about this option in the description of the `mmqos throttle create` command.

list
Lists the throttle system objects in the file system. You can specify either `--pool` or `--class` but not both. If you specify neither option, the command lists the throttle system objects for all pools and classes.

Device
The device name of the file system.

--pool *PoolName*
A storage pool. The command lists only the throttles that are configured to monitor the specified pool.

--class *ClassName*
A QoS class. The command lists only the throttles that are created for the specified class.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

config
Controls QoS attributes that apply at the file system level.

Note: To work with the same QoS attributes at the class level, see the descriptions of the following commands:

mmqos **class set**
mmqos **class list**

set *Device Attribute=Value*[**,***Attribute=Value***...**]
Sets the initial value to be assigned to an attribute of a class when the class is created. For example, the following command causes the class attribute `fileset-stats` to be initialized to `yes` in each subsequent new class. This command does not alter the value of `fileset-stats` in existing classes:

```
mmqos config set Device fileset-stats=yes
```

If the initial value is later set to `no`, as in the following command, then `fileset-stats` is initialized to `no` in each subsequent new class. This command does not affect the value of `fileset-stats` in existing classes:

```
mmqos config set Device fileset-stats=no
```

You cannot set the value `yes` for **pid-stats** and **fileset-stats** attributes at a time. That is, either **pid-stats=yes** or **fileset-stats=yes** can be used in the configuration.

The following table shows the initial values of all the file-system-level attributes when QoS is initialized:

Table 31. Attributes at the file system level

Attribute	Value when QoS is initialized
fileset-stats	**no**
fine-stats	0
mdio-enabled	**yes**
pid-stats	**no**
skim-factor	.987
stat-poll-interval	5
stat-slot-time	1000

Device
The device name of the file system.

***Attribute=Value*[*,Attribute=Value*...]**
The name of the attribute and the value to set for the attribute. The following attributes can be set:

fileset-stats={yes | no}
Specifies whether QoS keeps statistics for each fileset that is associated with the class. The default value is no, which means that QoS does not keep statistics.

fine-stats=*Seconds*
Specifies how many seconds of fine-grained statistics to save in memory so that the `mmqos report list` command can list them. The default value is 0, which means that QoS does not save any fine-grained statistics. The valid range is 0 - 3840 seconds.

Fine-grained statistics are collected at specified intervals and contain more information than regular statistics. The interval at which fine-grained statistics are collected is set with the `slat-slot-time` file-system-level attribute. The display of fine-grained statistics is controlled by the `--fine-stats-display-range` option of the `mmqos report list` command.

mdio-enabled={yes | no}
Enables or disables MDIO. When MDIO is enabled for a class, QoS dynamically balances the allocation of IOPS or MB/s among the nodes that have mounted the file system, based on the relative proportions of their attempted I/O accesses to the storage pool of the class. When MDIO is disabled for a class, QoS allocates a fixed number of IOPS or MB/s among the nodes that have mounted the file system and never changes the allocation.

By default, MDIO is enabled for QoS user classes and is disabled for QoS system classes. You can change the value of this attribute for individual classes with the `mmqos class set` command.

For more information, see the subtopic "Mount dynamic I/O (MDIO)" earlier in this topic.

pid-stats= {yes | no}
Specifies whether QoS keeps statistics for each process that accesses the storage pool of a QoS class. The default value is no, which means that QoS does not keep statistics for each process.

skim-factor=*Fraction*
This attribute applies only to QoS system classes. If throttles from two or more QoS classes are operating and one class has a surplus of unused I/O capacity, QoS appropriates or "skims" the unused I/O capacity and assigns it to other classes that need more I/O capacity. The `skim-factor` attribute specifies the fraction of I/O capacity that cannot be skimmed from the class by other classes. The default value is .987, which means that only 1.3 percent of the I/O capacity of the class can be skimmed by other classes.

It is a good idea not to modify the value of this attribute without a strong reason.

stat-poll-interval=*Seconds*
Specifies the interval in seconds between each sending of accumulated QoS statistics by a node to the QoS manager node. The QoS manager node is an internally selected node that stores and manages the statistics of a QoS class so that the statistics can be displayed by an `mmqos report list` command. The QoS statistics are described in the following subtopics, which appear later in this topic:

“Analyzing regular output from the mmqos report list command” on page 639
“Analyzing fine-grained output from the mmqos report list command” on page 640

At the beginning of each `stat-poll-interval`, the QoS component on the node sets the node's QoS statistics variables to zero. During the `stat-poll-interval` interval, the QoS component adds an amount to the appropriate statistics variable whenever a relevant event occurs. For example, when a process on a node consumes some number of IOPS, the QoS component adds the IOPS count to the statistics variable for the number of IOPS consumed by the node. At the end of the interval, the QoS component on the node sends the accumulated value for QoS statistics to the QoS manager node.

The valid range of values for `stat-poll-interval` is 0 - 120 seconds in one-second intervals. The default value is 5 seconds, which means that a node collects QoS statistics for 5 seconds before it sends the accumulated QoS statistics to the QoS manager node. The `stat-poll-interval` must be a multiple of the value of the `stat-slot-time` attribute. For more information, see the description of that attribute.

Reducing the frequency of QoS statistics reports:

For any QoS class, the number of statistics reports that the QoS manager node must process depends on the following factors:

The number of nodes that mount the file system
As the number of nodes increases, the number of reports also increases.

The value of `stat-poll-interval`
As the value decreases, the frequency of reports increases. In other words, the shorter that the data collection interval is, the greater are the number of reports that are sent per second or per minute to the QoS manager node.

The value of `stat-slot-time`
The `stat-slot-time` attribute provides a way to reduce the frequency of statistics reports that are sent to the QoS manager node. The value of `stat-slot-time` sets the length of the delay that QoS interposes between the end of one `stat-poll-interval` and the beginning of the next. As the value of `stat-slot-time` increases, the frequency of reports decreases. In other words, the greater that the delay is between reporting intervals, the fewer are the reports that are sent per second or per minute to the QoS manager node. For more information, see the description of `stat-slot-time` attribute that appears immediately after the description of `stat-slot-time` in this topic.

If the flow of new reports becomes too great, the QoS manager node can no longer update its stored statistics values in a timely manner. In that case, the statistics that are reported by the `mmqos report list` command can be incorrect.

Two options are available to compensate for the increase in the number of reports that occurs when the number of nodes that mount the file system increases by a significant amount:

- Activate the dynamic adjustment of `stat-poll-interval` and `stat-slot-time` by setting both attributes to zero. To suspend dynamic adjustment, set the value of either attribute to a nonzero value. For more information, see the subtopic "QoS statistics" on page 626 earlier in this topic.
- Manually increase the values of `stat-poll-interval` and `stat-slot-time`. For guidance in selecting values for `stat-poll-interval` and `stat-slot-time`, see Table 30 on page 627.

`stat-slot-time` *Milliseconds*
Specifies the amount of time in milliseconds that QoS waits after the end of one `stat-poll-interval` before starting the next. Valid values are 125, 250, 500, 1000, and 2000 - 24000 in 1000 millisecond increments. The default value is 125. Increasing the value of `stat-slot-time` reduces the number of QoS statistics reports that are sent to the QoS manager node.

`list`
Lists the current settings of the QoS data collection attributes for this file system.

Note: You can issue this command action early in your configuration process to see default data collection attribute values that have already been set. The following example shows the settings after `mmqos filesystem enable` *`Device`* is issued, where *Device* is `gpfs1`:

```
# mmqos config list gpfs1
 Configuration Option            Configuration Value
-------------------------------------------------
 fine-stats                      0
 pid-stats                       no
```

```
stat-poll-interval              5
stat-slot-time                  100
fileset-stats                   no
skim-factor                     0
```

All the settings that are listed in this example are default values.

Device
The device name of the file system.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

filesystem
Commands for controlling QoS processing for the specified file system.

init
Initializes the QoS environment for the specified file system. QoS initializes itself automatically if you issue a command that sets a configuration value, such as `mmqos class create`.

Device
The device name of the file system.

enable
Enables QoS processing for the specified file system. QoS begins or resumes regulating I/O operations and accumulating I/O performance statistics. This command does not modify any customer-created QoS configuration and does not discard accumulated performance statistics.

Device
The device name of the file system.

disable
Disables QoS processing for the specified file system. QoS stops regulating I/O operations and stops accumulating performance statistics. This command does not modify any customer-created QoS configuration and does not discard accumulated performance statistics.

Device
The device name of the file system.

reset
Removes all QoS customer-created QoS configuration information and accumulated performance statistics from the specified file system. Resets the QoS configuration to the same state that `mmqos filesystem init` sets. This command fails if QoS is in the enabled state.

Warning: Issue this command only if you are sure that you no longer need QoS functions in this file system. If you remove all the QoS configuration information and later find that you need QoS functions, you must re-create the configuration step-by-step.

Device
The device name of the file system.

refresh
Refreshes the active QoS configuration in the IBM Spectrum Scale daemon from the master QoS configuration that is stored in the CCR.

Device
The device name of the file system.

list [-Y]
Lists the file systems in the current cluster that are configured for QoS. Notice that this command does not require a device name as the first option.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

status [-Y]
Displays the current status of QoS for the specified file system.

Device
The device name of the file system.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

The following example displays the QoS status for file system `fs1`:

```
# mmqos filesystem status fs1
                  Configuration Repository        Daemon
                  ------------------------        ------
State:            disabled                        disabled
Version:          49                              49
Throttling:       --                              inactive
Monitoring:       --                              inactive
```

The command displays QoS status information for the file system both as it exists in the CCR and as it exists in the file system:

State
Indicates whether QoS is initialized, enabled, disabled, or reset.

Version
Shows how many configuration changes have occurred since the first QoS configuration action was done for this file system.

Throttling
Indicates whether any throttle rules are created.

Note: In `mmqos` output the value "`--`" indicates that the measurement is not applicable in the current context. In the preceding example, it indicates that throttling and monitoring do not occur in the CCR environment.

Monitoring
Indicates whether performance statistics are being collected.

restore --restore-file ***RestoreFile***
Restores the QoS configuration. Do not issue this command except with the guidance of IBM Support for a disaster recovery for QoS configuration data.

Note: This function requires that the `mmsdrbackup` user exit for the CCR has been configured to ensure that there is a backup that includes the QoS data as part of the CCR backup.

report
Commands for working with reports.

list
Writes a report to STDOUT.

Device
The device name of the file system.

--pool {all | *PoolName*}
The storage pools for which statistics are to be listed.

all
Lists statistics for all the pools that are assigned to QoS filesets.

PoolName
Lists statistics for the specified storage pool.

--seconds *Seconds*
Lists the I/O performance values that were collected during the previous specified number of seconds. The valid range is 1 - 999 seconds. The default is 60 seconds. The values are listed over a number of subperiods within the period that you specify. You cannot configure the number or length of subperiods. Examples of subperiods are every 5 seconds over the last 60 seconds or every 60 seconds over the last 600 seconds.

--sum-classes {yes | no}

yes
Lists the sum of the I/O performance values of all the QoS classes.

no
Lists the I/O performance of each QoS class separately. This value is the default.

--sum-nodes {yes |no}

yes
Lists the sum of the I/O performance values of all the nodes. This value is the default.

no
Lists the I/O performance of each node separately.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

--fine-stats-display-range *FineStatsDisplayRange*
Lists the fine-grained statistics that are currently available in memory. The *FineStatsDisplayRange* values specify the indexes of the first and last blocks of statistics in the range of blocks that you want to list, such as `0-4`. If you specify only one integer, such as 2, the command lists all the blocks in memory beginning with that index and going to the end. The last line of the output lists the index of the next block in memory. You can avoid relisting fine-grained statistics by beginning the next report at this block index.

Fine-grained statistics are taken at one-second intervals and contain more information than regular statistics. The default interval is one second. You can set the interval with the `stat-slot-time` option. They are intended to be used as input for programs that analyze and display data, such as the example plotting program at `/usr/lpp/mmfs/samples/charts/qosplotfine.pl`. For the content of the statistics, see the subtopic later in this topic.

For more information, see the subtopics "Analyzing regular output from the `mmqos report list` command" and "Analyzing fine-grained output from the `mmqos report list` command" later in this topic.

Exit status

0
Successful completion.

Nonzero
A failure occurred.

Security

You must have root authority to run the `mmqos` command.

The node on which you enter the command must be able to run remote shell commands on any other administration node in the cluster. It must be able to do so without the use of a password and without producing any extraneous messages. For more information, see the topic *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

Analyzing regular output from the `mmqos report list` command

The `mmqos report list` command always displays the following lines of output first:

QOS config::
This line indicates whether QoS is actively regulating I/O accesses (enabled) or not (disabled).

QOS values::
This line displays information for each pool that is part of a throttle configuration. For each pool, the information consists of the pool name followed information about each QoS class that accesses the pool. For each class, the information includes the IOPS or MB/s of the throttles and in some cases the throttle scope. In the following example fragment, the `QOS values` line shows that the system storage pool is used by a throttle for the `other` class that is set to 500 IOPS and a throttle for the `maintenance` class that is set to 200 MB/s. The line also displays the throttle scope of the `maintenance` class, `all_local`, which is the default throttle scope for the class:

```
QOS values:: pool=system,other=500IOps,maintenance/all_local=200MBs
```

Note: The `QOS values` line does not display a system class if the IOPS and MB/s of the class are set to `unlimited`.

QOS status::
This line indicates whether QoS is regulating the consumption of IOPS ("throttling") and also whether QoS is recording ("monitoring") the consumption of IOPS of each storage pool.

The following example shows the complete output of an `mmqos report list` command:

```
# mmqos report list gpfs1 --seconds 30
QOS config::     enabled -- --ccr:ccr-file-version=3
QOS values::     pool=system,other=100Iops
QOS status::     throttling active, monitoring active
QOS IO stats for pool: system
03:22:25 misc        iops=2.80000         MBs=0.06172      maxIOPS=unlimited   maxMBS=unlimited ioql=0.00136    qsdl=0.00000    et=5 wideOpen=yes(1:1:1)
03:22:25 other       iops=50.00000        MBs=197.60469    maxIOPS=50.0        maxMBS=unlimited ioql=3.04175    qsdl=22.94107   et=5 wideOpen=no(0:0:0)
03:22:30 other       iops=49.40000        MBs=197.60000    maxIOPS=50.0        maxMBS=unlimited ioql=2.99324    qsdl=27.92655   et=5 wideOpen=no(0:0:0)
03:22:35 misc        iops=2.20000         MBs=0.04219      maxIOPS=unlimited   maxMBS=unlimited ioql=0.00058    qsdl=0.00000    et=5 wideOpen=yes(1:1:1)
03:22:35 other       iops=43.20000        MBs=170.40469    maxIOPS=50.0        maxMBS=unlimited ioql=1.67852    qsdl=17.50021   et=5 wideOpen=no(0:0:0)
03:22:40 misc        iops=0.80000         MBs=0.00859      maxIOPS=unlimited   maxMBS=unlimited ioql=0.00031    qsdl=0.00000    et=5 wideOpen=yes(1:1:1)
03:23:10 misc        iops=1.20000         MBs=0.81484      maxIOPS=unlimited   maxMBS=unlimited ioql=0.00905    qsdl=0.00000    et=5 wideOpen=yes(1:1:1)
03:23:20 misc        iops=0.20000         MBs=0.00078      maxIOPS=unlimited   maxMBS=unlimited ioql=0.00006    qsdl=0.00000    et=5 wideOpen=yes(1:1:1)
mmqos: Command completed.
```

The command requests a list of I/O performance values for all QoS pools over the previous 30 seconds. Because the options `--sum_classes` and `--sum_nodes` are missing, the command also requests I/O performance for each storage pool separately and summed across all the nodes of the cluster.

Only one pool is part of a throttle configuration, the system storage pool. The output indicates that IOPS and MB/s occurred only for processes in the `other` class and in the `misc` class. The meaning of the categories in each line is as follows:

First column
The time when the measurement period ends.

Second column
The QoS class for which the measurement is made.

`iops=`
The performance of the class in I/O operations per second.

`MBs=`
The performance of the class in megabytes per second.

maxIOPS=
The maxIOPS throttling limitation for the interval.

maxMBS=
The maxMBS throttling limitation for the interval.

ioql=
The average number of I/O requests in the class that are pending for reasons other than being queued by QoS. This number includes, for example, I/O requests that are waiting for network or storage device servicing.

qsdl=
The average number of I/O requests in the class that are queued by QoS. When the QoS system receives an I/O request from the file system, QoS first finds the class to which the I/O request belongs. It then finds whether the class has any I/O operations available for consumption. If not, then QoS queues the request until more I/O operations become available for the class. The `Qsdl` value is the average number of I/O requests that are held in this queue.

et=
The interval in seconds during which the measurement was made.

wideOpen=
Whether the throttling of the class in that interval is unlimited (1) or not (0).

You can calculate the average service time for an I/O operation as `((Ioql + Qsdl) / Iops)`. For a system that is running IO-intensive applications, you can interpret the value `(Ioql + Qsdl)` as the number of threads in the I/O-intensive applications. This interpretation assumes that each thread spends most of its time in waiting for an I/O operation to complete.

Analyzing fine-grained output from the `mmqos report list` command

The following code block shows an example of fine-grained output. Each line displays the sum of the data for all the QoS programs that are running on the specified node. If the `--pid-stats` option is specified in the `mmqos report list` command, then each line displays the data for one QoS program that is running on the specified node:

```
Time, Class, Node, Iops, TotSctrs, Pool, Pid, RW, SctrI, AvgTm, SsTm, MinTm, MaxTm, AvgQd, SsQd
1480606227,misc,172.20.0.21,285,2360,system,0,R,16,0.000260,0.000197,0.000006,0.005894,0.000001,0.000000
1480606228,misc,172.20.0.21,631,5048,system,0,R,16,0.000038,0.000034,0.000006,0.002463,0.000000,0.000000
1480606239,other,172.20.0.21,13,112,system,26724,R,16,0.000012,0.000000,0.000008,0.000022,0.000001,0.000000
1480606239,other,172.20.0.21,1,512,system,26724,R,512,0.000375,0.000000,0.000375,0.000375,0.000002,0.000000
1480606239,other,172.20.0.21,30,15360,system,26724,W,512,0.000910,0.000004,0.000451,0.002042,0.000001,0.000000
1480606240,misc,172.20.0.21,5,48,system,14680072,W,16,0.000135,0.000000,0.000025,0.000258,0.000000,0.000000
1480606240,other,172.20.0.21,4,32,system,26724,R,16,0.000017,0.000000,0.000009,0.000022,0.000001,0.000000
1480606240,other,172.20.0.21,48,24576,system,26724,W,512,0.000916,0.000006,0.000490,0.002141,0.000001,0.000000
1480606241,other,172.20.0.21,10,80,system,26724,R,16,0.000017,0.000000,0.000007,0.000039,0.000001,0.000000
1480606241,other,172.20.0.21,34,17408,system,26724,W,512,0.000873,0.000007,0.000312,0.001957,0.000001,0.000000
1480606241,misc,172.20.0.21,12,104,system,15597572,W,16,0.000042,0.000000,0.000024,0.000096,0.000000,0.000000
1480606241,misc,172.20.0.21,1,512,system,15597572,W,512,0.000192,0.000000,0.000192,0.000192,0.000000,0.000000
1480606241,misc,172.20.0.21,9,72,system,14680071,W,16,0.000029,0.000000,0.000024,0.000040,0.000000,0.000000
## conti=4
```

The following table shows the same output in table format. Only the first 10 columns are shown:

Time	Class	Node	Iops	TotSctrs	Pool	Pid	RW	Sctrl	AvgTime
1480606227	misc	172.20.0.21	285	2360	system	0	R	16	0.00026
1480606228	misc	172.20.0.21	631	5048	system	0	R	16	0.000038
1480606239	other	172.20.0.21	13	112	system	26724	R	16	0.000012
1480606239	other	172.20.0.21	1	512	system	26724	R	512	0.000375
1480606239	other	172.20.0.21	30	15360	system	26724	W	512	0.00091
1480606240	misc	172.20.0.21	5	48	system	14680072	W	16	0.000135
1480606240	other	172.20.0.21	4	32	system	26724	R	16	0.000017
1480606240	other	172.20.0.21	48	24576	system	26724	W	512	0.000916
1480606241	other	172.20.0.21	10	80	system	26724	R	16	0.000017

Time	Class	Node	Iops	TotSctrs	Pool	Pid	RW	Sctrl	AvgTime
1480606241	other	172.20.0.21	34	17408	system	26724	W	512	0.000873
1480606241	misc	172.20.0.21	12	104	system	15597572	W	16	0.000042
1480606241	misc	172.20.0.21	1	512	system	15597572	W	512	0.000192
1480606241	misc	172.20.0.21	9	72	system	14680071	W	16	0.000029

The following list describes the type of content in each column:

Time
The time when the measurement was made, expressed in seconds since the UNIX epoch of January 1, 1970.

Class
The QoS class of the process.

Node
The IP address of the node on which the process was running.

Iops
The number of IOPS that the process consumed during the sample period.

MB/s
The number of MB/s that the process consumed during the sample period.

TotSctrs
The number of sectors for which data was read or written. By default, one sector is 512 bytes.

Note: The number of **TotSctrs** from one process or one node might not be equal to the actual number of bytes that are read or written. For cached I/O in IBM Spectrum Scale, if the modified data from one application I/O operation is less than **fgdbRangeSize** (4 KiB by default) IBM Spectrum Scale writes the number of bytes that are specified by **fgdbRangeSize** from the page pool to backend disks. If the application updates only one byte of the file, the value that is displayed by **TotSctrs** is eight sectors.

Pool
The storage pool where the I/O operations were done.

Pid
The process ID of the process that initiated the I/O. When the `--pid-stats` option of the `mmqos report list` command is not specified, this value is always 0.

RW
The type of I/O operation, read or write.

SctrI
The number of sectors that were affected by the I/O operation. This value is expressed in terms of one of the following measures:

A single sector.
1/32 or less of a full block.
Less than a full block.
A full block.

For example, if the disk block size is 512, the command displays one of the following values: 1, 16, 511, or 512.

AvgTm
The mean time that was required for an I/O operation to be completed.

SsTm
The sum of the squares of differences from the mean value that is displayed for `AvgTm`. You can use this value to calculate the variance of I/O service times.

MinTm
The minimum time that was required for an I/O operation to be completed.

MaxTm
The maximum time that was required for an I/O operation to be completed.

AvgQd
The mean time for which QoS imposed a delay of the read or write operation.

SsQd
The sum of the squares of differences from the mean value that is displayed for AvgQd.

The last line in the example indicates that the index of the next block to be displayed is 4:

```
## conti=4
```

You can avoid redisplaying statistics by having the next `mmqos report list` command display statistics beginning at this block index.

Examples

1. This example shows the steps for creating a new QoS configuration:

   a. Issue the `mmqos class create` command to create a user class `HR_Dept_1` and to associate it with an existing fileset `testfs1`:

   ```
   # mmqos class create gpfs1 --class "HR_Dept_1" --fileset testfs1
   mmqos: Processing the new QOS configuration...
   mmqos: Validating configuration changes with the daemon...

   mmqos: Validation complete.
   mmqos: The configuration data was written to the CCR successfully.
   mmqos: Calling the daemon to update the configuration within the cluster...

   QOS configuration has been installed and broadcast to all nodes.

   mmqos: Daemon update is complete.
   mmqos: Command completed.
   ```

   b. Optionally, you can Issue the `mmqos class list` command to verify that user class `HR_Dept_1` has been created and is associated with fileset `testfs1`:

   ```
   # mmqos class list gpfs1
   Class Name            Type     Fileset Name
   ----------------------------------------
   maintenance           system   --
   other                 system   --
   misc                  system   --
   mdio-all-sharing      system   --
   HR_Dept_1             user     testfs1
   ```

   c. Issue `mmqos throttle create` command to create a throttle. In this example, the command does the following actions:

      i) It creates a new throttle.

      ii) It associates the throttle with an existing storage pool C and with the new user class HR_Dept_1.

      iii) It assigns 100 IOPS to the throttle. When user-launched processes access storage pool C they will collectively be able to consume up to 100 I/0 operations per second.

   ```
   # mmqos throttle create gpfs1 --pool C --class HR_Dept_1 --maxiops 100
   mmqos:  Processing the new QOS configuration...
   mmqos:  Validating configuration changes with the daemon...

   Adjusted QOS Class specification: pool=C,HR_Dept_1=100Iops,maintenance/
   all_local=inf,other=inf,_skimf=0

   mmqos:  Validation complete.
   mmqos:  The configuration data was written to the CCR successfully.
   mmqos:  Calling the daemon to update the configuration within the cluster...

   QOS configuration has been installed and broadcast to all nodes.
   ```

```
mmqos:  Daemon update is complete.
mmqos:  Completed.
```

d. Optionally, issue the `mmqos throttle list` command to verify that the throttle has been created:

```
# mmlsqos throttle list gpfs1
 Pool Name     Class Name       Node/Nodeclass  Cluster Scope      MaxIOPS    MaxMBS
-------------------------------------------------------------------------------------
 C             HR_Dept_1        -               all                unlimited  100
```

e. Issue the `mmqos filesystem enable` command to enable QoS processing for file system `gpfs1`:

```
# mmqos filesystem enable gpfs1
mmqos:  Processing the new QOS configuration...
mmqos:  Validating configuration changes with the daemon...

Adjusted QOS Class specification: pool=C,HR_Dept_1=100Iops,maintenance/
all_local=inf,other=inf,_skimf=0

mmqos:  Validation complete.
mmqos:  The configuration data was written to the CCR successfully.
mmqos:  Calling the daemon to update the configuration within the cluster...

QOS configuration has been installed and broadcast to all nodes.

mmqos:  Daemon update is complete.
mmqos:  Completed.
```

The QoS class specification provides the following information:

- The only storage pool listed is storage pool `C`. That is because pool `C` is the only storage pool that has been associated with a throttle in file system `gpfs1`.
- When processes that belong to class `HR_Dept_1` access storage pool `C`, they can collectively consume up to 100 I/O operations per second. This category includes almost all user-launched processes, including most user-launched IBM Spectrum Scale administrative commands, such as `mmchfs`, `mmlsfs`, and many others.
- When processes that belong to the `maintenance` class access storage pool `C`, they can collectively consume an unlimited number of I/O operations per second. This category includes most of the long-running IBM Spectrum Scale administrative commands. For more information about this class, see section "QoS user classes" on page 621 earlier in this topic.
- When processes that belong to the `other` class access storage pool `C`, they can collectively consume an unlimited number of I/O operations per second. This category includes some critical file system processes. For more information about this class, see section "QoS user classes" on page 621 earlier in this topic.
- The skim factor is set to the default value which is 0.

f. Optionally, issue the `mmqos filesystem status` command to list the QoS status for file system `gpfs1`

```
#mmqos filesystem status gpfs1
                    Configuration Repository          Daemon
                    ------------------------          ------
State:              enabled                           enabled
Version:            3                                 3
Throttling:         --                                active
Monitoring:         --                                active

Configured System Objects:
--------------------------
QOS File Systems:         1
System Classes:           2
User Classes:             1
Filesets:                 1
Throttles:                1
```

g. Issue the `mmqos config set` command to cause QoS to collect statistics for every process that accesses pool C:

```
#mmqos config set  gpfs1  pid-stats=yes
mmqos:  Processing the new QOS configuration...

mmqos:  Processing config option 'pid-stats'
mmqos:  Validating configuration changes with the daemon...

Adjusted QOS Class specification: pool=C,HR_Dept_1=100Iops,maintenance/
all_local=inf,other=inf,_skimf=0

mmqos:  Validation complete.
mmqos:  The configuration data was written to the CCR successfully.
mmqos:  Calling the daemon to update the configuration within the cluster...

QOS configuration has been installed and broadcast to all nodes.

mmqos:  Daemon update is complete.
mmqos:  Completed.
```

h. Optionally, issue the `mmqos config list` command to display the current configuration for collecting QoS statistics for file system `gpfs1`. Because only `pid-stats` has been modified, the values that are shown for the other configuration settings are the default values:

```
# mmqos config list gpfs1
 Configuration Option            Configuration Value
-----------------------------------------------------
 fine-stats                      0         <default>
 pid-stats                       no        <default>
 stat-poll-interval              5         <default>
 stat-slot-time                  1000      <default>
 fileset-stats                   no        <default>
 skim-factor                     .987      <default>
 mdio-enabled                    yes       <default>
```

2. The following example shows how to issue the `mmqos report list` command to get QoS status information for a file system and QoS statistics information for a storage pool. In this example, the file system is `gpfs1` and the pool is `C`:

```
# mmqos report list gpfs1 --pool C
QOS config::    enabled -- --ccr:ccr-file-version=4
QOS values::    pool=system,other=inf,maintenance/all_local=inf:
                pool=A,other=inf,maintenance/all_local=inf:
                pool=B,other=inf,maintenance/all_local=inf:
                pool=C,HR_Dept_1=100Iops,maintenance/all_local=inf,other=inf,
                _skimf=0
QOS status:     throttling active, monitoring active
=== for pool C
01:50:45 misc iops=0.6 MBs=0.0019531 ioql=0.0004632 qsdl=4e-07 et=5
mmqos: Command completed.
```

3. The following example shows how to limit the I/O activity of the long running IBM Spectrum Scale maintenance commands that belong to the QoS `maintenance` system class. The `mmqos class list` command shows that only the four QoS system classes exist. No QoS user classes have been created:

```
# mmqos class list gpfs1
 Class Name                      Class Type    Fileset Name
-------------------------------------------------------------
 maintenance                     system        --
 other                           system        --
 misc                            system        --
 mdio-all-sharing                system        --
```

The following `mmqos throttle create` command creates a throttle for the `maintenance` class with an I/O limit of 200 MB/s. The command does not specify the `-N` or `-C` option, so QoS sets the scope of the new throttle to the default throttle scope value for the `maintenance` class, which is `all_local`:

```
# mmqos throttle create gpfs1 --pool system --class maintenance --maxMBS 200
mmqos: Detected the -N and -C options were not optionally supplied with the Maintenance
class.
```

```
mmqos: Processing the new QoS configuration...
mmqos: File System Manager QoS service validating the configuration (host is
gpfsnode1.gpfs.net)...

mmqos: Validation complete.
mmqos: The configuration data was written to the CCR successfully
mmqos: Calling the daemon to update the configuration within the cluster...

QOS configuration has been installed and broadcast to all nodes.

mmqos: Daemon update is complete.
mmqos: Command completed.
```

The `mmqos throttle list` command shows that the throttle has been created:

```
# mmqos throttle list gpfs1
 Pool Name     Class Name     Node/Nodeclass    Cluster Scope      MaxIOPS    MaxMBS
-------------------------------------------------------------------------------------
 system        maintenance    -                 all_local          unlimited  200
```

This example assumes that QoS is already enabled in the file system, so it is not necessary to issue the `mmqos filesystem enable` command to start the new throttle working. The last step is to change the I/O limit of the throttle to 400 MB/s:

```
# mmqos throttle update gpfs1 --pool system --class maintenance -C all_local --maxMBS 400
mmqos: Processing the new QoS configuration...
mmqos: File System Manager QoS service validating the configuration (host is
gpfsnode1.gpfs.net)...

mmqos: Validation complete.
mmqos: The configuration data was written to the CCR successfully
mmqos: Calling the daemon to update the configuration within the cluster...

QOS configuration has been installed and broadcast to all nodes.

mmqos: Daemon update is complete.
mmqos: Command completed.
```

The `mmqos throttle list` command shows that the maximum MB/s setting has changed to 400:

```
# mmqos throttle list gpfs1
 Pool Name     Class Name     Node/Nodeclass    Cluster Scope      MaxIOPS    MaxMBS
-------------------------------------------------------------------------------------
 system        maintenance    -                 all_local          unlimited  400
```

4. The following example shows how to create a QoS user class and modify the throttle scope and the I/O limits:

   a. The following `mmqos class list` command shows that no QoS user classes have been created yet:

   ```
   # mmqos class list gpfs1
    Class Name                    Class Type    Fileset Name
   -----------------------------------------------------------
    maintenance                   system        --
    other                         system        --
    misc                          system        --
    mdio-all-sharing              system        --
   ```

   b. The following `mmqos class create` command creates the QoS user class `HR_Dept_2`, which includes filesets `hrfset1` and `hrfset2`:

   ```
   # mmqos class create gpfs1 --class HR_Dept_2 --fileset hrfset1,hrfset2
   mmqos: Processing the new QoS configuration...
   mmqos: File System Manager QoS service validating the configuration (host is
   gpfsnode1.gpfs.net)...

   mmqos: Validation complete.
   mmqos: The configuration data was written to the CCR successfully
   mmqos: Calling the daemon to update the configuration within the cluster...

   QOS configuration has been installed and broadcast to all nodes.
   ```

```
mmqos: Daemon update is complete.
mmqos: Command completed.
```

The `mmqos class list` command shows that the user class `HR_Dept_2` is created:

```
# mmqos class list gpfs1
 Class Name                     Class Type    Fileset Name
-----------------------------------------------------------
 maintenance                    system        --
 other                          system        --
 misc                           system        --
 mdio-all-sharing               system        --
 HR_Dept_2                      user          hrfset1,hrfset2
```

c. A throttle is created for the new QoS user class. The `--force` option is used to set the I/O limit below 100 IOPS to 90 IOPS:

```
# mmqos throttle create gpfs1 --pool system --class HR_Dept_2 --maxiops 90 --force
mmqos: Processing the new QoS configuration...
mmqos: File System Manager QoS service validating the configuration (host is
gpfsnode1.gpfs.net)...


mmqos: Validation complete.
mmqos: The configuration data was written to the CCR successfully
mmqos: Calling the daemon to update the configuration within the cluster...

QOS configuration has been installed and broadcast to all nodes.

mmqos: Daemon update is complete.
mmqos: Command completed.
```

The `mmqos throttle list` command shows that the throttle is created:

```
# mmqos throttle list gpfs1
 Pool Name      Class Name     Node/Nodeclass     Cluster Scope      MaxIOPS    MaxMBS
-------------------------------------------------------------------------------------
 system         maintenance    -                  all_local          unlimited  200
 system         HR_Dept_2      -                  -                  90         unlimited
```

d. The `mmqos throttle update` command is issued to change the maximum IOPS setting of the throttle to 300 IOPS:

```
# mmqos throttle update gpfs1 --pool system --class HR_Dept_2 --maxiops 300
mmqos: Processing the new QoS configuration...
mmqos: File System Manager QoS service validating the configuration (host is
gpfsnode1.gpfs.net)...

mmqos: Validation complete.
mmqos: The configuration data was written to the CCR successfully
mmqos: Calling the daemon to update the configuration within the cluster...

QOS configuration has been installed and broadcast to all nodes.

mmqos: Daemon update is complete.
mmqos: Command completed.
```

The `mmqos throttle list` command shows that the maximum IOPS for the throttle is now 300 IOPS:

```
# mmqos throttle list gpfs1
 Pool Name      Class Name     Node/Nodeclass     Cluster Scope      MaxIOPS    MaxMBS
-------------------------------------------------------------------------------------
 system         maintenance    -                  all_local          unlimited  200
 system         HR_Dept_2      -                  -                  300        unlimited
```

e. The `mmqos throttle updatekey` command is issued to narrow the scope of the throttle to a single node `gpfsnode2.gpfs.net`:

```
# mmqos throttle updatekey  gpfs1 --pool system --class HR_Dept_2  --new-N
gpfsnode2.gpfs.net
mmqos: Processing the new QoS configuration...
mmqos: File System Manager QoS service validating the configuration (host is
gpfsnode1.gpfs.net)...
```

```
mmqos: Validation complete.
mmqos: The configuration data was written to the CCR successfully
mmqos: Calling the daemon to update the configuration within the cluster...

QOS configuration has been installed and broadcast to all nodes.

mmqos: Daemon update is complete.
mmqos: Command completed.
```

The `mmqos throttle list` command shows that the scope of the throttle is now set to node `gpfsnode2.gpfs.net`:

```
# mmqos throttle list gpfs1
 Pool Name    Class Name     Node/Nodeclass         Cluster Scope     MaxIOPS    MaxMBS
-------------------------------------------------------------------------------------
 system       maintenance    -                      all_local         unlimited  200
 system       HR_Dept_2      gpfsnode2.gpfs.net     -                 300        unlimited
```

f. The `mmqos throttle update` command is issued to change the maximum MB/s I/O limit to 400 MB/s. Notice that the new throttle scope `-N gpfsnode2.gpfs.net` must be specified explicitly to enable QoS to identify the throttle:

```
# mmqos throttle update gpfs1 --pool system --class HR_Dept_2 -N gpfsnode2.gpfs.net --
maxMBS 400
mmqos: Processing the new QoS configuration...
mmqos: File System Manager QoS service validating the configuration (host is
gpfsnode1.gpfs.net)...


mmqos: Validation complete.
mmqos: The configuration data was written to the CCR successfully
mmqos: Calling the daemon to update the configuration within the cluster...

QOS configuration has been installed and broadcast to all nodes.

mmqos: Daemon update is complete.
mmqos: Command completed.
```

The `mmqos throttle list` command shows that the throttle now has a maximum IOPS setting of 300 IOPS and a maximum MB/s setting of 400 MB/s:

```
# mmqos throttle list gpfs1
 Pool Name      Class Name     Node/Nodeclass      Cluster Scope      MaxIOPS    MaxMBS
-----------------------------------------------------------------------------------------
 system         maintenance    -                    all_local          unlimited  200
 system         HR_Dept_2      gpfsnode2.gpfs.net   -                  300        400
```

g. Finally, the `mmqos throttle delete` command is issued to delete the throttle:

```
# mmqos throttle delete gpfs1 --pool system --class HR_Dept_2 -N gpfsnode2.gpfs.net
mmqos: Processing the new QoS configuration...
mmqos: File System Manager QoS service validating the configuration (host is
gpfsnode1.gpfs.net)...

mmqos: Validation complete.
mmqos: The configuration data was written to the CCR successfully
mmqos: Calling the daemon to update the configuration within the cluster...

QOS configuration has been installed and broadcast to all nodes.

mmqos: Daemon update is complete.
mmqos: Command completed.
```

The `mmqos throttle list` command no longer lists the QoS user throttle, because it has been deleted:

```
# mmqos throttle list gpfs1
 Pool Name      Class Name     Node/Nodeclass      Cluster Scope      MaxIOPS    MaxMBS
-----------------------------------------------------------------------------------------
 system         maintenance    -                    all_local          unlimited  200
```

5. The following example shows how to create a throttle for the `mdio-all-sharing` class to configure class sharing within the system pool and QoS user classes `HR_Dept_1` and `HR_Dept_2`. Initially the

`mmqos class list` command shows only the four system classes and the two user classes, each of which includes a fileset:

```
# mmqos class list gpfs1
 Class Name                     Class Type    Fileset Name
-----------------------------------------------------------
 maintenance                    system        --
 other                          system        --
 misc                           system        --
 mdio-all-sharing               system        --
 HR_Dept_2                      user          fstest2
 HR_Dept_1                      user          fstest1
```

The `mmqos throttle list` command lists a throttle for the `maintenance` class and a throttle for each of the two user classes. The pool for all three throttles is the system storage pool:

```
# mmqos throttle list gpfs1
 Pool Name     Class Name     Node/Nodeclass    Cluster Scope      MaxIOPS    MaxMBS
-------------------------------------------------------------------------------------
 system        maintenance    -                 all_local          unlimited  400
 system        HR_Dept_1      -                 -                  unlimited  200
 system        HR_Dept_2      -                 -                  unlimited  300
```

The following `mmqos throttle create` command creates a throttle for the `mdio-all-sharing` system class with an I/O limit of 500 MB/s:

```
# mmqos throttle create gpfs1 --pool system --class mdio-all-sharing --maxmbs 500
mmqos: Processing the new QoS configuration...
mmqos: File System Manager QoS service validating the configuration (host is
gpfsnode1.gpfs.net)...

mmqos: Validation complete.
mmqos: The configuration data was written to the CCR successfully
mmqos: Calling the daemon to update the configuration within the cluster...

QOS configuration has been installed and broadcast to all nodes.

mmqos: Daemon update is complete.
mmqos: Command completed.
```

The `mmqos throttle list` command shows that the throttle for the `mdio-all-sharing` class is created:

```
# mmqos throttle list gpfs1
 Pool Name     Class Name          Node/Nodeclass    Cluster Scope      MaxIOPS    MaxMBS
-------------------------------------------------------------------------------------------
 system        maintenance         -                 all_local          unlimited  400
 system        HR_Dept_1           -                 -                  unlimited  200
 system        HR_Dept_2           -                 -                  unlimited  300
 system        mdio-all-sharing    -                 -                  unlimited  500
```

6. If one of the system objects that can be elements of a QoS class or a QoS throttle, such as a fileset, a pool, a node, a node class, or a cluster, is deleted before the class or throttle is deleted, the configuration of the class or throttle contains references to non-existent objects. These references can cause the QoS configuration validation phase to fail and display error messages when you try to revise or delete the class or throttle.

   To revise or delete the class or throttle, you can use the special option `QOSSkipValidate=1` to override the QoS configuration validation phase. This option must be used only in this situation. Using the option in any other situation can result in an invalid configuration.

   The following two examples illustrate how to use this option. In the first example, the fileset `hrfset1`, which is included in QoS user class `HR_Dept_2`, was accidentally deleted from the system with the command `mmdelfileset`. When you try to remove the fileset from the class, the command fails with a configuration validation error. To remove the fileset from the class, issue the following command:

```
#  QOSSkipValidate=1  mmqos class update gpfs1 --class HR_Dept_2 --remove --fileset hrfset1
mmqos: Processing the new QoS configuration...
mmqos: The configuration data was written to the CCR successfully
mmqos: Calling the daemon to update the configuration within the cluster...
```

```
QOS configuration has been installed and broadcast to all nodes.

mmqos: Daemon update is complete.
mmqos: Command completed.

# mmqos class list gpfs1
 Class Name                     Class Type    Fileset Name
-----------------------------------------------------------
 maintenance                    system        --
 other                          system        --
 misc                           system        --
 mdio-all-sharing               system        --
 HR_Dept_2                      user          hrfset2
```

In the second example, the node `gpfsnode1.gpfs.net`, which defines the throttle scope of QoS user class `HR_Dept_2`, was removed from the cluster. When you try to delete the throttle, the command fails with a configuration validation error. To delete the throttle, issue the following command:

```
#  QOSSkipValidate=1  mmqos  throttle delete gpfs1 --pool system --class HR_Dept_2 -N
gpfsnode1.gpfs.net
mmqos: Processing the new QoS configuration...
mmqos: The configuration data was written to the CCR successfully
mmqos: Calling the daemon to update the configuration within the cluster...

QOS configuration has been installed and broadcast to all nodes.

mmqos: Daemon update is complete.
mmqos: Command completed.
```

7. The following example shows how to issue the `mmqos filesystem reset` command to reset all QoS configuration information for a file system to the default values. Follow these steps:

   a. Issue the `mmqos filesystem reset` command to reset all the configuration information for file system `gpfs1`:

   ```
   # mmqos filesystem reset gpfs1

   mmqos: Are you certain you want to remove all specific QOS configuration for this device
   (gpfs1)?
          This action will reset all QoS configuration to default values.
          This action cannot be undone.

          Permanently remove all specific QoS configuration data for device gpfs1 (yes/no)?
   yes
   QOS configuration has been installed and broadcast to all nodes
   ```

   Note:

   - Before you do this action, save any QoS configuration information that you might want to use again either with this file system or with another file system. Issue `mmqos` commands to display the current configuration information. Then copy the information to a secure file.
   - Instead of resetting the QoS configuration information to default values, consider issuing the `mmqos filesystem disable` command to disable all QoS activity for the file system. When you are ready, issue the `mmqos filesystem enable` command to enable QoS activity for the file system.

   b. Optionally, issue the `mmqos filesystem status` command to verify that the QoS configuration information has been reset:

   ```
   # mmqos filesystem status gpfs1
                      Configuration Repository        QoS Daemon Service
                      ------------------------        ------------------
   State:             disabled                        disabled
   Version:           3                               3
   Throttling:        --                              inactive
   Monitoring:        --                              inactive

   Configured System Objects:
   --------------------------
   QoS File Systems:        3

   Class Objects:
   ```

```
  System Classes:        4
  User Classes:          0
  Filesets:              0

Throttle Objects:        0
```

mmquotaoff command

Deactivates quota limit checking.

Synopsis

```
mmquotaoff [-u] [-g] [-j] [-v] {Device [Device ...] | -a}
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmquotaoff` command disables quota limit checking by GPFS.

If none of: `-u`, `-j` or `-g` is specified, the `mmquotaoff` command deactivates quota limit checking for users, groups, and filesets.

If the `-a` option is not specified, *Device* must be the last parameter entered.

Parameters

***Device* [*Device* ...]**
The device name of the file system to have quota limit checking deactivated.

If more than one file system is listed, the names must be delimited by a space. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

Options

-a
Deactivates quota limit checking for all GPFS file systems in the cluster. When used in combination with the `-g` option, only group quota limit checking is deactivated. When used in combination with the `-u` or `-j` options, only user or fileset quota limit checking, respectively, is deactivated.

-g
Specifies that only group quota limit checking is to be deactivated.

-j
Specifies that only quota checking for filesets is to be deactivated.

-u
Specifies that only user quota limit checking is to be deactivated.

-v
Prints a message for each file system in which quotas are deactivated.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmquotaoff` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

GPFS must be running on the node from which the `mmquotaoff` command is issued.

Examples

1. To deactivate user quota limit checking on file system `fs0`, issue this command:

```
mmquotaoff -u fs0
```

   To confirm the change, issue this command:

```
mmlsfs fs0 -Q
```

   The system displays information similar to:

```
flag value          description
---- -------------- ---------------
 -Q  group;fileset  Quotas enforced
```

2. To deactivate group quota limit checking on all file systems, issue this command:

```
mmquotaoff -g -a
```

   To confirm the change, individually for each file system, issue this command:

```
mmlsfs fs2 -Q
```

   The system displays information similar to:

```
flag value          description
---- -------------- -----------------
 -Q  user;fileset   Quotas enforced
```

3. To deactivate all quota limit checking on file system `fs0`, issue this command:

```
mmquotaoff fs0
```

   To confirm the change, issue this command:

```
mmlsfs fs0 -Q
```

   The system displays information similar to:

```
flag value          description
---- -------------- ----------------
 -Q  none           Quotas enforced
```

See also

- “mmcheckquota command” on page 220
- “mmdefedquota command” on page 345
- “mmdefquotaoff command” on page 349
- “mmdefquotaon command” on page 352
- “mmedquota command” on page 404
- “mmlsquota command” on page 535
- “mmquotaon command” on page 654
- “mmrepquota command” on page 666

Location

/usr/lpp/mmfs/bin

mmquotaon command

Activates quota limit checking.

Synopsis

```
mmquotaon [-u] [-g] [-j] [-v] {Device [Device...] | -a}
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The mmquotaon command enables quota limit checking by GPFS.

If none of: -u, -j or -g is specified, the mmquotaon command activates quota limit checking for users, groups, and filesets.

If the -a option is not used, *Device* must be the last parameter specified.

After quota limit checking has been activated by issuing the mmquotaon command, issue the mmcheckquota command to count inode and space usage.

Parameters

***Device* [*Device*...]**
The device name of the file system to have quota limit checking activated.

If more than one file system is listed, the names must be delimited by a space. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

Options

-a
Activates quota limit checking for all of the GPFS file systems in the cluster. When used in combination with the -g option, only group quota limit checking is activated. When used in combination with the -u or -j option, only user or fileset quota limit checking, respectively, is activated.

-g
Specifies that only group quota limit checking is to be activated.

-j
Specifies that only fileset quota checking is to be activated.

-u
Specifies that only user quota limit checking is to be activated.

-v
Prints a message for each file system in which quota limit checking is activated.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the mmquotaon command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

GPFS must be running on the node from which the `mmquotaon` command is issued.

Examples

1. To activate user quotas on file system `fs0`, issue this command:

```
mmquotaon -u fs0
```

   To confirm the change, issue this command:

```
mmlsfs fs0 -Q
```

   The system displays information similar to:

```
flag value          description
---- -------------- ----------------
 -Q  user           Quotas enforced
```

2. To activate group quota limit checking on all file systems, issue this command:

```
mmquotaon -g -a
```

   To confirm the change, individually for each file system, issue this command:

```
mmlsfs  fs1 -Q
```

   The system displays information similar to:

```
flag value          description
---- -------------- ----------------
 -Q  group          Quotas enforced
```

3. To activate user, group, and fileset quota limit checking on file system `fs2`, issue this command:

```
mmquotaon fs2
```

   To confirm the change, issue this command:

```
mmlsfs  fs2 -Q
```

   The system displays information similar to:

```
flag value               description
---- --------------      ---------------
 -Q  user;group;fileset  Quotas enforced
```

See also

- “mmcheckquota command” on page 220
- “mmdefedquota command” on page 345
- “mmdefquotaoff command” on page 349
- “mmdefquotaon command” on page 352
- “mmedquota command” on page 404
- “mmlsquota command” on page 535
- “mmquotaoff command” on page 651
- “mmrepquota command” on page 666

Location

/usr/lpp/mmfs/bin

mmreclaimspace command

Reclaims free space on a GPFS file system.

Synopsis

```
mmreclaimspace Device [-Y] [-P PoolName] [-qos QosClass]
                  {--reclaim-threshold Percentage | --emergency-reclaim}
```

Availability

Available on all IBM Spectrum Scale editions.

Space reclamation

For more information about space reclamation in IBM Spectrum Scale and a list of supported operating systems and verified storage systems, see the topic *IBM Spectrum Scale with data reduction storage devices* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Description

Use the `mmreclaimspace` command to reclaim unused device sectors in a GPFS file system. This command is useful for solid-state drives (SSDs) and thin provisioned storage. Reclaiming means informing the device that data that is saved on these sectors is no longer used by the file system. These sectors can be reused by the device for other purposes, such as improving internal storage management in SSDs or reusing the space for this or another volume.

Note: The `mmreclaimspace` command is effective only if the file system includes at least one SSD or thin provisioned disk. For more information, see the topic *IBM Spectrum Scale with data reduction storage devices* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

For each disk in the GPFS file system, the `mmreclaimspace` command displays the following information, by failure group and by storage pool:

- The size of the disk.
- The failure group of the disk.
- Whether the disk is used to hold data, metadata, or both.
- Available space in full blocks.
- Available space in subblocks.
- Reclaimed space in subblocks. This value indicates the number of subblocks that are discarded on the device by this instance of the command.

Displayed values are rounded down to a multiple of 1024 bytes. If the subblock size that is used by the file system is not a multiple of 1024 bytes, then the displayed values can be lower than the actual values. This situation can result in the display of a total value that exceeds the sum of the rounded values that are displayed for individual disks. The individual values are accurate if the subblock size is a multiple of 1024 bytes.

The `mmreclaimspace` command can be run against a mounted or unmounted file system.

Note:

- This command is I/O-intensive and should be run only when the system load is light.
- An asterisk at the end of a line indicates that the disk is in a state where it is not available for new block allocation.

Parameters

Device
The device name of the file system to be queried for available file space. File system names need not be fully qualified. `fs0` is as acceptable as `/dev/fs0`.

This parameter must be the first parameter in the command.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

-P ***PoolName***
Reclaim space only for disks that belong to the specified storage pool.

--qos ***QosClass***
Specifies the Quality of Service for I/O operations (QoS) class to which the instance of the command is assigned. If you do not specify this parameter, the instance of the command is assigned by default to the `maintenance` QoS class. This parameter has no effect unless the QoS service is enabled. For more information, see the topic "mmchqos command" on page 263. Specify one of the following QoS classes:

maintenance
This QoS class is typically configured to have a smaller share of file system IOPS. Use this class for I/O-intensive, potentially long-running GPFS commands, so that they contribute less to reducing overall file system performance.

other
This QoS class is typically configured to have a larger share of file system IOPS. Use this class for administration commands that are not I/O-intensive.

For more information, see the topic *Setting the Quality of Service for I/O operations (QoS)* in *IBM Spectrum Scale: Administration Guide*.

--reclaim-threshold ***Percentage***
Specifies the threshold of reclaimable space, as a percentage value, beyond which a block allocation segment will be selected to be reclaimed. A percentage of 0 means that all segments that have reclaimable space are selected for reclaiming; a percentage of 90 means that only segments that have reclaimable space over 90% are selected for reclaiming.

Considering that the block allocation segment is exclusively locked while space on it is being reclaimed, it is a good practice to choose a bigger value, such as 90, if the application is latency-sensitive, and choose a smaller value, such as 0, to reclaim all reclaimable space if the system is almost idle. The block allocation segment is a logical concept that is used to manage the disk space in a GPFS file system. Each segment contains free space and reclaimable space information.

--emergency-reclaim
Performs emergency recovery and releases pre-reserved space for the next GPFS file system recovery. If the back-end storage is thin provisioned, it is possible that writing to a file in a GPFS file system might fail because the thin-provisioned storage is out of physical space. That situation can result in the file system being unmounted and not being recovered. To bring the file system back into operation, an emergency recovery procedure is necessary. Issuing the `mmreclaimspace` command with the `--emergency-reclaim` option is part of the recovery process.

Note: This option works only if the thin-provisioned storage is read-writable and the file system is mounted in restrict mode.

For more information, see the topic *IBM Spectrum Scale with data reduction storage devices* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Exit status

0
Successful completion.

Nonzero
A failure has occurred.

Security

You must have root authority to run the `mmreclaimspace` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

To reclaim space on all the space reclaimable disks in the `foofs` file system, issue the following command:

```
mmreclaimspace foofs --reclaim-threshold 0
```

The command displays information as in the following example:

```
disk           disk size  failure holds    holds           free in KB    free in KB  reclaimed in KB
name               in KB    group metadata data        in full blocks  in fragments     in subblocks
---------------------- -------- -------- ----- -------------------- ------------- ----------------
Disks in storage pool: system (Maximum disk size allowed is 8.82 TB)
gpfs1nsd      1169920000       -1 Yes      No       1162211328 ( 99%)   15896 ( 0%)  1162227224 (99%)
            ------------                           -------------------- ------------- ----------------
(pool total)  1169920000                          1162211328 ( 99%)   15896 ( 0%)  1162227224 (99%)

Disks in storage pool: data (Maximum disk size allowed is 8.93 TB)
gpfs2nsd      1169920000        1 No       Yes      1161445376 ( 99%)   20320 ( 0%)  1161465696 (99%)
gpfs3nsd      1169920000        2 No       Yes      1161445376 ( 99%)   20320 ( 0%)  1161465696 (99%)
            ------------                           -------------------- ------------- ----------------
(pool total)  2339840000                          2322890752 ( 99%)   40640 ( 0%)  2322931392 (99%)

            ============                           ==================== =============
(data)        2339840000                          2322890752 ( 99%)   40640 ( 0%)
(metadata)    1169920000                          1162211328 ( 99%)   15896 ( 0%)
            ============                           ==================== ============= ================
(total)       3509760000                          3485102080 ( 99%)   56536 ( 0%)  3485158616 (99%)
```

See also

- “mmcrnsd command” on page 335
- “mmcrfs command” on page 318
- “mmchdisk command” on page 212
- “mmadddisk command” on page 28
- “mmrpldisk command” on page 690
- “mmlsfs command” on page 506
- *IBM Spectrum Scale with data reduction storage devices* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Location

`/usr/lpp/mmfs/bin`

mmremotecluster command

Manages information about remote GPFS clusters.

Synopsis

```
mmremotecluster add RemoteClusterName [-n ContactNodes] [-k KeyFile]
```

or

```
mmremotecluster update RemoteClusterName [-C NewClusterName] [-n ContactNodes] [-k KeyFile]
```

or

```
mmremotecluster delete {RemoteClusterName | all}
```

or

```
mmremotecluster show [RemoteClusterName | all] [-Y]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmremotecluster` command is used to make remote GPFS clusters known to the local cluster, and to maintain the attributes associated with those remote clusters. The keyword appearing after `mmremotecluster` determines which action is performed:

add
Adds a remote GPFS cluster to the set of remote clusters known to the local cluster.

delete
Deletes the information for a remote GPFS cluster.

show
Displays information about a remote GPFS cluster.

update
Updates the attributes of a remote GPFS cluster.

To be able to mount file systems that belong to some other GPFS cluster, you must first make the nodes in this cluster aware of the GPFS cluster that owns those file systems. This is accomplished with the `mmremotecluster add` command. The information that the command requires must be provided to you by the administrator of the remote GPFS cluster. You will need this information:

- The name of the remote cluster.
- The names or IP addresses of a few nodes that belong to the remote GPFS cluster.
- The public key file generated by the administrator of the remote cluster by issuing the `mmauth genkey` command for the remote cluster.

Since each cluster is managed independently, there is no automatic coordination and propagation of changes between clusters like there is between the nodes within a cluster. This means that once a remote cluster is defined with the `mmremotecluster` command, the information about that cluster is automatically propagated across all nodes that belong to this cluster. But if the administrator of the remote cluster decides to rename it, or deletes some or all of the contact nodes, or change the public key file, the information in this cluster becomes obsolete. It is the responsibility of the administrator of the remote GPFS cluster to notify you of such changes so that you can update your information using the appropriate options of the `mmremotecluster update` command.

Parameters

RemoteClusterName
Specifies the cluster name associated with the remote cluster that owns the remote GPFS file system. The value `all` indicates all remote clusters defined to this cluster, when using the `mmremotecluster delete` or `mmremotecluster show` commands.

-C *NewClusterName*
Specifies the new cluster name to be associated with the remote cluster.

-k *KeyFile*
Specifies the name of the public key file provided to you by the administrator of the remote GPFS cluster.

-n *ContactNodes*
A comma separated list of nodes that belong to the remote GPFS cluster, in this format:

```
[tcpPort=NNNN,]node1[,node2 ...]
```

where:

```
tcpPort=NNNN
```

Specifies the TCP port number to be used by the local GPFS daemon when contacting the remote cluster. If not specified, GPFS will use the default TCP port number 1191.

```
node1[,node2...]
```

Specifies a list of nodes that belong to the remote cluster. The nodes can be identified through their host names or IP addresses.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

Exit status

0
Successful completion. After successful completion of the `mmremotecluster` command, the new configuration information is propagated to all nodes in the cluster.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmremotecluster` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. This command adds remote cluster `k164.kgn.ibm.com` to the set of remote clusters known to the local cluster, specifying k164n02 and k164n03 as remote contact nodes. File `k164.id_rsa.pub` is the name of the public key file provided to you by the administrator of the remote cluster.

```
mmremotecluster add k164.kgn.ibm.com -n k164n02,k164n03 -k k164.id_rsa.pub
```

The output is similar to this:

```
mmremotecluster: 6027-1371 Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

2. This command displays information for the remote cluster k164.kgn.ibm.com.

```
mmremotecluster show k164.kgn.ibm.com
```

The output is similar to this:

```
Cluster name:    k164.kgn.ibm.com
Contact nodes:   k164n02,k164n03
SHA digest:      a3917c8282fca7a27d951566940768dcd241902b
File systems:    (none defined)
```

For more information on the SHA digest, see the *IBM Spectrum Scale: Problem Determination Guide* and search on *SHA digest*.

3. This command updates information for the remote cluster k164.kgn.ibm.com, changing the remote contact nodes to k164n02 and k164n01. The TCP port to be used when contacting cluster k164.kgn.ibm.com is defined to be 6667.

```
mmremotecluster update k164.kgn.ibm.com -n tcpPort=6667,k164n02,k164n01
```

The output is similar to this:

```
mmremotecluster: 6027-1371 Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

The `mmremotecluster show` command can then be used to see the changes.

```
mmremotecluster show k164.kgn.ibm.com
```

The output is similar to this:

```
Cluster name:    k164.kgn.ibm.com
Contact nodes: tcpPort=6667,k164n02,k164n01
SHA digest:      a3917c8282fca7a27d951566940768dcd241902b
File systems:    (none defined)
```

For more information on the SHA digest, see the *IBM Spectrum Scale: Problem Determination Guide* and search on *SHA digest*.

4. This command deletes information for remote cluster k164.kgn.ibm.com from the local cluster.

```
mmremotecluster delete k164.kgn.ibm.com
```

The output is similar to this:

```
mmremotecluster: 6027-1371 Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

See also

- “mmauth command” on page 96
- “mmremotefs command” on page 663

See also the topic about accessing GPFS file systems from other GPFS clusters in the *IBM Spectrum Scale: Administration Guide*.

Location

/usr/lpp/mmfs/bin

mmremotefs command

Manages information needed for mounting remote GPFS file systems.

Synopsis

```
mmremotefs add Device -f RemoteDevice -C RemoteClusterName
              [-T MountPoint] [-t DriveLetter]
              [-A {yes | no | automount}] [-o MountOptions] [--mount-priority Priority]
```

or

```
mmremotefs delete {Device | all | -C RemoteClusterName} [--force]
```

or

```
mmremotefs show [Device | all | -C RemoteClusterName] [-Y]
```

or

```
mmremotefs update Device [-f RemoteDevice] [-C RemoteClusterName]
              [-T MountPoint] [-t DriveLetter]
              [-A {yes | no | automount}] [-o MountOptions] [--mount-priority Priority]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmremotefs` command is used to make GPFS file systems that belong to other GPFS clusters known to the nodes in this cluster, and to maintain the attributes associated with these file systems. The keyword appearing after `mmremotefs` determines which action is performed:

add
Define a new remote GPFS file system.

delete
Delete the information for a remote GPFS file system.

show
Display the information associated with a remote GPFS file system.

update
Update the information associated with a remote GPFS file system.

Use the `mmremotefs` command to make the nodes in this cluster aware of file systems that belong to other GPFS clusters. The cluster that owns the given file system must have already been defined with the `mmremotecluster` command. The `mmremotefs` command is used to assign a local name under which the remote file system will be known in this cluster, the mount point where the file system is to be mounted in this cluster, and any local mount options that you may want.

Once a remote file system has been successfully defined and a local device name associated with it, you can issue normal commands using that local name, the same way you would issue them for file systems that are owned by this cluster.

When running the `mmremotefs` command delete and update options, the file system must be unmounted on the local cluster. However, it can be mounted elsewhere.

Parameters

Device
Specifies the name by which the remote GPFS file system will be known in the cluster.

-C *RemoteClusterName*
Specifies the name of the GPFS cluster that owns the remote GPFS file system.

-f *RemoteDevice*
Specifies the actual name of the remote GPFS file system. This is the device name of the file system as known to the remote cluster that owns the file system.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

Options

-A {yes | no | automount}
Indicates when the file system is to be mounted:

yes
When the GPFS daemon starts.

no
Manual mount. This is the default.

automount
When the file system is first accessed.

-o *MountOptions*
Specifies the mount options to pass to the mount command when mounting the file system. For a detailed description of the available mount options, see *GPFS-specific mount options* in *IBM Spectrum Scale: Administration Guide*.

-T *MountPoint*
The local mount point directory of the remote GPFS file system. If it is not specified, the mount point will be set to *DefaultMountDir*/*Device*. The default value for *DefaultMountDir* is `/gpfs`, but it can be changed with the `mmchconfig` command.

-t *DriveLetter*
Specifies the drive letter to use when the file system is mounted on Windows.

--mount-priority *Priority*
Controls the order in which the individual file systems are mounted at daemon startup or when one of the `all` keywords is specified on the `mmmount` command.

File systems with higher *Priority* numbers are mounted after file systems with lower numbers. File systems that do not have mount priorities are mounted last. A value of zero indicates no priority.

--force
The --force flag can only be used with the delete option. It will override an error that can occur when trying to delete a remote mount where the remote cluster was already removed. If the original delete attempt returns an error stating it cannot check to see if the mount is in use, then this is the condition to use. The --force flag overrides and allows the deletion to complete.

Exit status

0
Successful completion. After successful completion of the `mmremotefs` command, the new configuration information is propagated to all nodes in the cluster.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmremotefs` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see the topic *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

Examples

This command adds remote file system `gpfsn`, owned by remote cluster `k164.kgn.ibm.com`, to the local cluster, assigning `rgpfsn` as the local name for the file system, and `/gpfs/rgpfsn` as the local mount point.

```
mmremotefs add rgpfsn -f gpfsn -C k164.kgn.ibm.com -T /gpfs/rgpfsn
```

The output is similar to this:

```
mmremotefs: 6027-1371 Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

The `mmremotefs show` command can be used to see the changes.

```
mmremotefs show rgpfsn
```

The output is similar to this:

```
Local Name  Remote Name  Cluster name               Mount Point      Mount Options
Automount  Drive
rgpfs1      gpfs1        gpfs-n60-win.fvtdomain.net /rgpfs1          rw
no         K
```

See also

- “mmauth command” on page 96
- “mmremotecluster command” on page 660

See also the topic about accessing GPFS file systems from other GPFS clusters in the *IBM Spectrum Scale: Administration Guide*.

Location

`/usr/lpp/mmfs/bin`

mmrepquota command

Displays file system user, group, and fileset quotas.

Synopsis

```
mmrepquota [-u] [-g] [-e] [-n] [-v]
           [--block-size {BlockSize | auto} | -Y] {-a | Device:Fileset ...}
```

or

```
mmrepquota-j [-e] [-q] [-n] [-v] [-t]
           [--block-size {BlockSize | auto}] {-a | Device...}
```

or

```
mmrepquota[-q] [-t] {-a | Device...}
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmrepquota` command reports file system usage and quota information for a user, group, or fileset.

This command cannot be run from a Windows node.

Important: Quota limits are not enforced for root users (by default). For information on managing quotas, see *Managing GPFS quotas* in the *IBM Spectrum Scale: Administration Guide*.

If `-g`, `-j`, or `-u` are not specified, then user, group, and fileset quotas are listed.

If `-a` is not specified, *Device* must be the last parameter entered.

For each file system in the cluster, the `mmrepquota` command displays:

1. Block limits (displayed in number of data blocks in 1 KB units or a unit that is defined by the `--block-size` parameter):
   - Quota type (USR, GRP, or FILESET)
   - Current usage (the amount of disk space used by this user, group, or fileset, in 1 KB units or a unit defined by the `--block-size` parameter)
   - Soft limit (the amount of disk space that this user, group, or fileset is allowed to use during normal operation, in 1 KB units or a unit defined by the `--block-size` parameter)
   - Hard limit (the total amount of disk space that this user, group, or fileset is allowed to use during the grace period, in 1 KB units or a unit defined by the `--block-size` parameter)
   - Space in doubt
   - Grace period
2. File limits:
   - Current number of files
   - Soft limit
   - Hard limit
   - Files in doubt
   - Grace period

Note: In cases where small files do not have an extra block that is allocated for them, quota usage might show less space usage than expected.

3. Entry Type

   default on
   Default quotas are enabled for this file system.

   default off
   Default quotas are not enabled for this file system.

   e
   Explicit quota limits are set by using the `mmedquota` command.

   d_fsys
   The quota limits are the default file system values set by using the `mmdefedquota` command.

   d_fset
   The quota limits are the default fileset-level values set by using the `mmdefedquota` command.

   i
   Default quotas were not enabled when this initial entry was established. Initial quota limits have a value of zero indicating no limit.

Because the sum of the in-doubt value and the current usage must not exceed the hard limit, the actual block space and number of files available to the user, group, or fileset might be constrained by the *in-doubt* value. If the *in-doubt* values approach a significant percentage of the quota, run the `mmcheckquota` command to account for the lost space and files.

For more information, see *Listing quotas* in the *IBM Spectrum Scale: Administration Guide*.

Parameters

Device
The device name of the file system to be listed.

If more than one file system is listed, the names must be delimited by a space. File system names need not be fully qualified. **fs0** is as acceptable as `/dev/fs0`.

Fileset
Specifies an optional fileset to be listed.

-a
Lists quotas for all file systems in the cluster. A header line is printed automatically with this option.

-e
Specifies that the `mmrepquota` command is to collect updated quota usage data from all nodes before displaying results. If this option is not specified, there is the potential to display negative usage values because the quota server might process a combination of up-to-date and back-level information.

-g
Lists only group quotas.

-j
Lists only fileset quotas.

-n
Displays a numerical user ID.

-q
Shows whether file system quota enforcement and default quota enforcement are active.

-t
Lists global user, group, and fileset block and inode grace times.

-u
Lists only user quotas.

-v
Prints a header line for the file systems that are being queried and adds an entryType description for each quota entry.

--block-size {*BlockSize* | auto}
Specifies the unit in which the number of blocks is displayed. The value must be of the form [*n*]K, [*n*]M, [*n*]G or [*n*]T, where *n* is an optional integer in the range 1 - 1023. The default is 1 K. If `auto` is specified, the number of blocks is automatically scaled to an easy-to-read value.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmrepquota` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see the topic *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

GPFS must be running on the node from which the `mmrepquota` command is issued.

Examples

1. To report on user quotas for file system `fs2` and display a header line, issue this command:

```
mmrepquota -u -v fs2
```

The system displays information similar to:

```
*** Report for USR quotas on fs2
              Block Limits               |           File Limits
                           in                                 in         entry
Name  type  KB quota limit doubt   grace |files quota limit doubt grace Type
root  USR    8     0     0     0    none |    1     0     0   0    none default on
user2 USR 2016   256   512     0   6days |    7    10    20   0    none d_fsys
user3 USR  104   256   512     0    none |    1    10    20   0    none d_fsys
user4 USR    0   256   512     0    none |    0    10    20   0    none d_fsys
user5 USR  368   256   512     0 23hours |    5     4    10   0 23hours d_fsys
user6 USR    0   256   512     0    none |    0    10    20   0    none d_fsys
user7 USR 1024  1024  5120  4096    none |    1     0     0  19    none e
```

2. To report on quota enforcement for `fs2`, issue this command:

```
mmrepquota -q fs2
```

The system displays information similar to:

```
fs2: USR quota is on; default quota is on
fs2: GRP quota is on; default quota is on
fs2: FILESET quota is on; default quota is off
```

3. To report on user quotas for file system `gpfs2`, issue this command:

```
mmrepquota -u gpfs2
```

The system displays information similar to:

```
         Block Limits                                |    File Limits
                                        in                                 in
Name   fileset type  KB quota  limit doubt grace | files quota limit doubt grace
root   root    USR    0     0      0     0  none |     1     0     0     0  none
```

```
root   fset3   USR    8     0      0     0  none |     1     0     0     0  none
root   fset4   USR 8192     0      0     0  none |     1     0     0     0  none
pfs001 root    USR    0 10240      0     0  none |     0  1000  5000     0  none
pfs001 fset3   USR    0 10240      0     0  none |     0  1000  5000     0  none
pfs001 fset4   USR 4104 10240 153600     0  none |     2  1000  5000     0  none
```

4. To report on user quotas for file system gpfs2 in fileset fset4, issue this command:

```
mmrepquota -u gpfs2:fset4
```

The system displays information similar to:

```
          Block Limits                                |        File Limits
                                          in                                  in
Name   fileset type   KB quota  limit doubt grace | files quota limit doubt grace
root   fset4   USR  8192     0      0     0  none |     1     0     0     0  none
pfs001 fset4   USR  4104 10240 153600     0  none |     2  1000  5000     0  none
```

5. To list global user, group, and fileset block and inode grace times, issue this command:

```
mmrepquota -u -t gpfs_s
```

The system displays information similar to:

```
User:    block default grace time 7days, inode default grace time 7days
Group:   block default grace time 7days, inode default grace time 7days
Fileset: block default grace time 7days, inode default grace time 7days
      Block Limits                                |     File Limits
Name  type      KB  quota  limit  in_doubt  grace | files  quota  limit  in_doubt  grace
root  USR        0      0      0         0   none |    50      0      0         0   none
ftp   USR        0      0      0         0   none |    50      0      0         0   none
```

6. To report on fileset quotas in file system fs1, issue this command:

```
mmrepquota -j fs1 --block-size auto
```

The system displays information similar to:

```
                              Block Limits                                |              File Limits
Name      fileset    type       blocks   quota   limit   in_doubt  grace  |  files   quota    limit
in_doubt    grace
root      root       FILESET      256K       0       0          0   none  |      1       0
0        0     none
fset0     root       FILESET    89.25G    100G    200G     89.99M   none  |  13729    4000
5000        0   7 days
fset1     root       FILESET         0    100G    200G          0   none  |      1    4000
5000        0     none
```

Note: In any **mmrepquota** listing, when the type is FILESET, the Name column heading is meant to indicate the fileset name (root, fset0, and fset1 in this example), and the value for the fileset column heading (root, in this example) can be ignored.

See also

- “mmcheckquota command” on page 220
- “mmdefedquota command” on page 345
- “mmdefquotaoff command” on page 349
- “mmdefquotaon command” on page 352
- “mmedquota command” on page 404
- “mmlsquota command” on page 535
- “mmquotaoff command” on page 651
- “mmquotaon command” on page 654

Location

/usr/lpp/mmfs/bin

mmrestoreconfig command

Restores file system configuration information.

Synopsis

```
mmrestoreconfig Device -i InputFile [-I {yes | test}]
          [-Q {yes | no | only}] [-W NewDeviceName]
```

or

```
mmrestoreconfig Device -i InputFile --image-restore
          [-I {yes | test}] [-W NewDeviceName]
```

or

```
mmrestoreconfig Device -i InputFile -F QueryResultFile
```

or

```
mmrestoreconfig Device -i InputFile -I continue
```

Availability

Available on all IBM Spectrum Scale editions. Available on AIX and Linux.

Description

The `mmrestoreconfig` command allows you to either query or restore, or both query *and* restore, the output file of the `mmbackupconfig` command.

In the **query phase**, the `mmrestoreconfig` command uses the output file generated by the `mmbackupconfig` command as an input parameter, and then creates a configuration file. Users can then edit the configuration file to fit their current file system configuration. You can use the definitions in the configuration file to create the appropriate network shared disks (NSDs) and file systems required for the restore.

In the **image restore phase**, the `mmrestoreconfig` command uses the input file (output from the `mmbackupconfig` command) to restore the backed up file system configuration in the newly created file system. The newly created file system must not be mounted prior to the `mmimgrestore` command execution thus the quota settings are turned off for image restore. They can be reactivated after the `mmimgrestore` command is completed using the `-Q only` flag of the `mmrestoreconfig` command.

This command cannot be run from a Windows node.

Parameters

Device
Specifies the name of the file system to be restored.

-i *inputFile*
Specifies the file generated by the `mmbackupconfig` command. The input file contains the file system configuration information.

-I {yes | test}
Specifies the action to be taken during the restore phase:

yes
Test and proceed on the restore process. This is the default action.

test
Test all the configuration settings before the actual restore is performed.

Use `-I continue` to restart `mmrestoreconfig` from the last known successful configuration restore.

-F *QueryResultFile*
Specifies the pathname of the configuration query result file generated by `mmrestoreconfig`. The configuration query result file is a report file that you can edit and use as a guide to `mmcrnsd` or `mmcrfs`.

--image-restore
Restores the configuration data in the proper format for Scale Out Backup and Restore (SOBAR).

-Q {yes | no | only}
Specifies whether quota settings are enforced during the file system restore. If set to no, the quota settings are ignored.

To restore quota settings after the `mmimgrestore` command has successfully run, the `-Q only` option must be specified.

-W *newDeviceName*
Restores the backed up file system information to this new device name

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmrestoreconfig` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. Run `mmrestoreconfig -F` *QueryResultFile* to specify the pathname of the configuration query result file to be generated.

```
mmrestoreconfig gpfs1 -i inputFile -F reportfile
```

2. To test settings before running a restore:

```
mmrestoreconfig fs1 -i /tmp/fs1.mmbackupconfig.out -I test
```

The system displays output similar to:

```
------------------------------------------------------------
Configuration test restore of fs1 begins at Wed Mar 14 16:00:16 EDT 2012.
------------------------------------------------------------
mmrestoreconfig: Checking disk settings for fs1:
mmrestoreconfig: Checking the number of storage pools defined for fs1.
The restored filesystem currently has 1 pools defined.
mmrestoreconfig: Checking storage pool names defined for fs1.
Storage pool 'system' defined.
mmrestoreconfig: Checking storage pool size for 'system'.
mmrestoreconfig: Storage pool size 127306752 was defined for 'system'.

mmrestoreconfig: Checking filesystem attribute configuration for fs1:
File system attribute to be restored: defaultDataReplicas
   Backup value:  2
   Current value: 1

mmrestoreconfig: Checking fileset configurations for fs1:
Fileset to restore: root.
```

```
Fileset status: Linked /fs1
Fileset mode: off
Fileset to restore: smallfileset.
Fileset status: Linked /fs1/smallfileset
Fileset mode: off

mmrestoreconfig: Checking policy rule configuration for fs1:
mmrestoreconfig: Testing policy configuration restore.
Validated policy `policyfile.backup': parsed 1 Placement Rules, 0 Restore Rules,
        0 Migrate/Delete/Exclude Rules,
        0 List Rules, 0 External Pool/List Rules

mmrestoreconfig: Checking quota settings for fs1:
mmrestoreconfig: Checking quota enablement for fs1.

mmrestoreconfig: Disabling the following settings:
mmrestoreconfig: Enabling the following settings:  -u -g -j

mmrestoreconfig: Disabling the following default quota settings:  -u -g -j

mmrestoreconfig: Enabling the following default quota settings:  -u -g -j

mmrestoreconfig: Quota limits for fs1:

mmrestoreconfig: Default Quota limits for fs1:

mmrestoreconfig: Command successfully completed
```

3. Run `mmrestoreconfig` to restore the **gpfs1** file system:

```
mmrestoreconfig gpfs1 -i inputFile
```

4. To restore the `fs9` file system configuration data prior to the `mmimgrestore` command, issue:

```
mmrestoreconfig fs9 -i fs9.backupconfig --image-restore
```

   The system displays output similar to:

```
mmrestoreconfig: Quota and DMAPI are enabled.
mmrestoreconfig: Disabling quota and/or DMAPI ...
----------------------------------------------------
Configuration restore of fs9 begins at Thu Nov 29 17:09:55 EST 2012.
----------------------------------------------------
mmrestoreconfig: Checking disk settings for fs9:
mmrestoreconfig: Checking the number of storage pools defined for fs9.
mmrestoreconfig: Checking storage pool names defined for fs9.
mmrestoreconfig: Checking storage pool size for 'system'.

mmrestoreconfig: Checking filesystem attribute configuration for fs9:

mmrestoreconfig: Checking policy rule configuration for fs9:
mmrestoreconfig: No policy rules installed in backed up filesystem fs9.
mmrestoreconfig: Command successfully completed
```

5. To restore the quota settings for file system `fs9`, after the `mmimgrestore` command, issue:

```
mmrestoreconfig fs9 -i fs9.backupconfig -Q only
```

   The system displays output similar to:

```
----------------------------------------------------
Configuration restore of fs9 begins at Thu Nov 29 17:13:51 EST 2012.
----------------------------------------------------

mmrestoreconfig: Checking quota settings for fs9:
mmrestoreconfig: Checking quota enablement for fs9.

mmrestoreconfig: Restoring quota and defquota limits for fs9:
fs9: Start quota check
  11 % complete on Thu Nov 29 17:17:37 2012
  22 % complete on Thu Nov 29 17:17:37 2012
  33 % complete on Thu Nov 29 17:17:37 2012
  44 % complete on Thu Nov 29 17:17:37 2012
  55 % complete on Thu Nov 29 17:17:38 2012
  69 % complete on Thu Nov 29 17:17:38 2012
  84 % complete on Thu Nov 29 17:17:38 2012
 100 % complete on Thu Nov 29 17:17:39 2012
```

```
Finished scanning the inodes for fs9.
Merging results from scan.
mmrestoreconfig: Command successfully completed
```

See also

- “mmbackupconfig command” on page 111
- “mmimgbackup command” on page 458
- “mmimgrestore command” on page 462

Location

/usr/lpp/mmfs/bin

mmrestorefs command

Restores a file system or an independent fileset from a snapshot.

Synopsis

```
mmrestorefs Device SnapshotName [-j FilesetName]
            [-N {Node[,Node...] | NodeFile | NodeClass}]
            [--log-quiet] [--preserve-encryption-attributes]
            [--suppress-external-attributes] [--threads MaxNumThreads]
            [--work-unit FilesPerThread]
```

Availability

Available on all IBM Spectrum Scale editions. Available on AIX and Linux.

Description

Use the `mmrestorefs` command to restore user data and attribute files to a file system or an independent fileset using those of the specified snapshot. Data will be restored by `mmrestorefs` without regard for file system or fileset quotas unless the `enforceFilesetQuotaOnRoot` configuration attribute of the `mmchconfig` command is set to `yes`. The `mmrestorefs` command does not restore the file system and fileset quota configuration information.

In versions before IBM Spectrum Scale 4.1.1, ensure that the file system is unmounted before you run the `mmrestorefs` command. When restoring from an independent fileset snapshot (using the `-j` option), link the fileset from nodes in the cluster that are to participate in the restore. It is preferable to run the `mmrestorefs` command when there are no user operations (either from commands, applications, or services) in progress on the file system or fileset. If there are user operations in progress on the file system or fileset while `mmrestorefs` is running, the restore might fail. For these failures, stop the user operations and run the `mmrestorefs` command again to complete the restore. For better performance, run the `mmrestorefs` command when the system is idle. While the restore is in progress, do not unlink the fileset, unmount the file system, or delete the fileset, fileset snapshot, or file system.

The `mmrestorefs` command cannot restore a fileset that was deleted after a global snapshot was created. In addition, the filesets in a global snapshot that are in deleted or unlinked state cannot be restored.

Snapshots are not affected by the `mmrestorefs` command. When a failure occurs during a restore, try repeating the `mmrestorefs` command except when there are ENOSPC or quota exceeded errors. In these cases, fix the errors then try the `mmrestorefs` command again.

For information on how GPFS policies and snapshots interact, see the *IBM Spectrum Scale: Administration Guide*.

Because snapshots are not copies of the entire file system, they should not be used as protection against media failures. For protection against media failures, see the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide* and search on "recoverability considerations".

The `mmrestorefs` command can cause a compressed file in the active file system to become decompressed if it is overwritten by the restore process. To recompress the file, run the `mmrestripefile` command with the `-z` option.

CAUTION:

- Do not run file compression or decompression while an `mmrestorefs` command is running. This caution applies to compression or decompression with the `mmchattr` command or with the `mmapplypolicy` command.
- Do not run the `mmrestripefs` or `mmrestripefile` command while an `mmrestorefs` command is running.

Parameters

Device
The device name of the file system that contains the snapshot to use for the restore. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

This must be the first parameter.

SnapshotName
Specifies the name of the snapshot that will be used for the restore.

-j *FilesetName*
Specifies the name of a fileset covered by this snapshot.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies the nodes that are to participate in the restore. The default is `all` or the current value of the `defaultHelperNodes` parameter of the `mmchconfig` command.

Starting with IBM Spectrum Scale 4.1.1, `-N` can be used for both fileset and global snapshot restores. (In GPFS 4.1, `-N` can be used for fileset snapshot restore only. In GPFS 3.5 and earlier, there is no `-N` parameter.)

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

--log-quiet
Suppresses detailed thread log output.

--preserve-encryption-attributes
Preserves the encryption extended attributes. Files that were removed after the snapshot was taken are restored with the same encryption attributes (including FEK) of the file in the snapshot. If this option is not used, the file is recreated with the encryption policy in place at the time the file is restored.

--suppress-external-attributes
Specifies that external attributes will not be restored.

--threads *MaxNumThreads*
Specifies the maximum number of concurrent restore operations. The default is 24.

--work-unit *FilesPerThread*
Specifies the number of files each thread will process at a time. The default is 100.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmrestorefs` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

Suppose that you have the following directory structure:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3
```

```
/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3
```

The directory `userA` is then deleted, leaving the following structure:

```
/fs1/file1

/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3
```

The directory `userB` is then created using the inode originally assigned to `userA`, and another snapshot is taken:

```
mmcrsnapshot fs1 snap2
```

The output is similar to this:

```
Writing dirty data to disk.
Quiescing all file system operations.
Writing dirty data to disk again.
Snapshot snap2 created with id 2.
```

The directory structure is similar to the following:

```
/fs1/file1
/fs1/userB/file2b
/fs1/userB/file3b

/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3

/fs1/.snapshots/snap2/file1
/fs1/.snapshots/snap2/userB/file2b
/fs1/.snapshots/snap2/userB/file3b
```

The file system is then restored from `snap1`:

```
mmrestorefs fs1 snap1
```

The resulting directory structure is similar to the following:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3

/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3

/fs1/.snapshots/snap2/file1
/fs1/.snapshots/snap2/userB/file2b
/fs1/.snapshots/snap2/userB/file3b
```

See also

- “mmcrsnapshot command” on page 340
- “mmdelsnapshot command” on page 383
- “mmlssnapshot command” on page 540
- “mmsnapdir command” on page 722

Location

`/usr/lpp/mmfs/bin`

mmrestripefile command

Rebalances or restores the replication factor of the specified files, performs any incomplete or deferred file compression or decompression, or detects and repairs data and directory block replica mismatches in the specified files.

Synopsis

```
mmrestripefile {-m | -r | -p | -b [--strict] | -l | -c [--read-only] | -z}
               {--inode-number [SnapPath/]InodeNumber [[SnapPath/]InodeNumber...] |
               --inode-number-file InodeNumberFile |
               -F FilenameFile | Filename [Filename...]}
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmrestripefile` command attempts to repair the specified files, performs any deferred or incomplete compression or decompression of the specified files, or detects and repairs data and directory block replica mismatches in the specified files. You can use `-F` option to specify a file that contains the list of file names to be processed or the `--inode-number-file` option to specify the list of inode numbers to be processed, with one file name per line.

The repair options are rebalancing (`-b`), restoring replication factors (`-r`), migrating data (`-m`), migrating file data to the proper pool (`-p`), relocating the block placement of the files (`-l`), data and directory block replica compare and repair (`-c`), and performing any deferred or incomplete compression or decompression (`-z`). The `-b` option not only rebalances files but also performs all the operations of the `-m` and `-r` options. For more information, see *Restriping a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

CAUTION: Do not run the `mmrestripefs` or `mmrestripefile` command while an `mmrestorefs` command is running.

Parameters

--inode-number [*SnapPath*/]*InodeNumber*
Specifies the inode number of the file to be restriped. If the current working directory is not already inside the active file system or snapshot, then the inode number has to be prefixed by the path to the active file system or snapshot. For example:

```
mmrestripefile –inode-number /fs0/.snapshots/snap1/34608
```

You must have root authority to use this option.

--inode-number-file *InodeNumberFile*
Specifies a file that contains a list of inode numbers of files to be restriped, one inode number per line. You must have root authority to use this option.

-F *FilenameFile*
Specifies a file that contains a list of names of files to be restriped, one name per line.

Filename
Specifies the names of one or more files to be restriped.

Options

stop

-m
Migrates critical data from any suspended disk for a list of specified files. Critical data is all data that would be lost if currently suspended disks were removed.

-r
Migrates all data for a list of files from suspended disks. If a disk failure or removal makes some replicated data inaccessible, this command also restores replicated files to their designated level of replication. Use this option immediately after a disk failure to protect replicated data against a subsequent failure. You can also use this option before you take a disk offline for maintenance to protect replicated data against the failure of another disk during the maintenance process.

-p
Moves the data of ill-placed files to the correct storage pool.

Some utilities, including the `mmchattr` command, can assign a file to a different storage pool without moving the data of the file to the new pool. Such files are called *ill-placed*. The **-p** parameter causes the command to move the data of ill-placed files to the correct storage pools.

-b [--strict]
Rebalances the specified files to improve file system performance. Rebalancing attempts to distribute file blocks evenly across the disks of the file system. In IBM Spectrum Scale 5.0.0 and later, rebalancing is implemented by a lenient round-robin method that typically runs faster than the previous method of strict round robin. To rebalance the file system using the strict round-robin method, include the `--strict` option.

--strict
Rebalances the specified files with a strict round-robin method. In IBM Spectrum Scale 4.2.3 and earlier versions, rebalancing always uses this method.

Note: Rebalancing distributes file blocks across all the disks in the cluster that are not suspended, including stopped disks. For stopped disks, rebalancing does not allow read operations and allocates data blocks without writing them to the disk. When the disk is restarted and replicated data is copied onto it, the file system completes the write operations.

-l
Relocates the block placement of the file. The location of the blocks depends on the current write affinity depth, write affinity failure group setting, block group factor, and the node from which the command is run. For example, for an existing file, regardless of how its blocks are distributed on disks currently, if `mmrestripefile -l` is run from node A, the final block distribution looks as if the file was created from scratch on node A.

To specify the write affinity failure group where the replica is put, before you run `mmrestripefile -l`, enter a command like the following example:

```
mmchattr --write-affinity-failure-group "WadfgValueString" filename
```

where *WadfgValueString* is the failure group.

-c [--read-only]
Compares the data or directory blocks of the specified files with their replicas and attempts to fix any mismatches. This option does not compare metadata blocks like indirect blocks. It also does not compare files whose size is less than the inode size, because their data is stored in the inode. To find the inode size, enter the following command:

```
mmlsfs <file_system_name> -i
```

--read-only
Only compares replicas and does not attempt to fix any mismatches.

-z
Performs any deferred or incomplete compression or decompression of files. For more information, see the topic *File compression* in the *IBM Spectrum Scale: Administration Guide*.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have write access to the file to run the `mmrestripefile` command unless you run with the `-c --read-only` option, in which case read access to the file is sufficient.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

This example illustrates restriping a file that is named `testfile0`. The following command confirms that the file is ill-placed:

```
mmlsattr -L testfile0
```

The system displays the following output:

```
file name:            testfile0
metadata replication: 2 max 2
data replication:     2 max 2
immutable:            no
appendOnly:           no
flags:                illplaced
storage pool name:    system
fileset name:         root
snapshot name:
```

To correct the problem, issue the following command:

```
mmrestripefile -p testfile0
```

To confirm the change, issue the following command:

```
mmlsattr -L testfile0
```

The system displays the following output:

```
file name:            testfile0
metadata replication: 2 max 2
data replication:     2 max 2
immutable:            no
appendOnly:           no
flags:
storage pool name:    system
fileset name:         root
snapshot name:
```

The following command compresses or decompresses a file for which compression or decompression is deferred or incomplete:

```
mmrestripefile -z  largefile.data
```

See also

- “mmadddisk command” on page 28
- “mmapplypolicy command” on page 80

- “mmchattr command” on page 157
- “mmchdisk command” on page 212
- “mmdeldisk command” on page 364
- “mmrpldisk command” on page 690
- “mmrestripefs command” on page 682

Location

`/usr/lpp/mmfs/bin`

mmrestripefs command

Rebalances or restores the replication factor of all the files in a file system. Alternatively, this command performs any incomplete or deferred file compression or decompression of all the files in a file system.

Synopsis

```
mmrestripefs Device {-m | -r | -b [--strict] | -R | -p | -z | --check-conflicting-replicas}
                    [-N {Node[,Node...] | NodeFile | NodeClass}] [-o InodeResultFile]
                    [-P PoolName] [--inode-criteria CriteriaFile]
                    [--pit-continues-on-error] [--qos QosClass]
```

or

```
mmrestripefs Device {-r | -b [--strict] | -R | --check-conflicting-replicas}
                    [-N {Node[,Node...] | NodeFile | NodeClass}] [-o InodeResultFile]
                    [--inode-criteria CriteriaFile] --metadata-only
                    [--pit-continues-on-error] [--qos QosClass]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Issue the `mmrestripefs` command to rebalance or restore the replication of all files in a file system. The command moves existing file system data between different disks in the file system based on changes to the disk state made by the `mmchdisk`, `mmadddisk`, and `mmdeldisk` commands. It also attempts to restore the metadata or data replication of all the files in the file system.

Tip: The `mmrestripefs` command can take a long time to run if there are many files or a large amount of data to rebalance or restore. If you are adding, deleting, or replacing multiple disks at the same time (`mmadddisk`, `mmdeldisk`, or `mmrpldisk`) you can run the `mmrestripefs` command after you have added, deleted, or replaced all the disks, rather than after each disk.

Alternatively, you can issue the `mmrestripefs` command to perform any deferred or incomplete file compression or decompression in all the files of a file system.

You must specify one of the options (`-m`, `-r`, `-b`, `-R`, `-p`, `-z`, or `--check-conflicting-replicas`) to indicate how much file system data to move or whether to perform file compression or decompression. You can issue this command against a mounted or unmounted file system.

If the file system uses replication, then restriping the file system also replicates it. Also, if the file system uses replication the `-r` option and the `-m` options treat suspended disks differently. The `-r` option removes all data from a suspended disk. But the `-m` option leaves data on a suspended disk if at least one replica of the data remains on a disk that is not suspended.

The `-b` option performs all the operations of the `-m` and `-r` options.

Use the `-z` option to perform any deferred or incomplete file compression or decompression.

CAUTION: Do not issue the `mmrestripefs` or `mmrestripefile` command while an `mmrestorefs` command is running.

Consider the necessity of restriping and the current demands on the system. New data that is added to the file system is correctly striped. Restriping a large file system requires many insert and delete operations and might affect system performance. Plan to perform this task when system demand is low.

Parameters

Device
The device name of the file system to be restriped. File system names need not be fully qualified.

Device must be the first parameter. It can take the following parameters:

-m
Migrates all critical data off any suspended disk in this file system. Critical data is all data that would be lost if currently suspended disks were removed.

-r
Migrates all data off suspended disks. It also restores all replicated files in the file system to their designated degree of replication when a previous disk failure or removal of a disk makes some replica data inaccessible. Use this parameter either immediately after a disk failure to protect replicated data against a subsequent failure, or before you take a disk offline for maintenance to protect replicated data against failure of another disk during the maintenance process.

Note: If the file system uses replication, before running **mmrestripefs Device -r**, you should run **mmlsdisk Device -L** to check the number of failure groups available. If the number of failure groups available is less than your default replication, you should not run **mmrestripefs Device -r** because it will remove the data replica for files that have replicas that are located on suspended or to be emptied disks.

-b [--strict]
Rebalances the file system to improve performance. Rebalancing attempts to distribute file blocks evenly across the disks of the file system. In IBM Spectrum Scale 5.0.0 and later, rebalancing is implemented by a lenient round-robin method that typically runs faster than the previous method of strict round robin. To rebalance the file system with the strict round-robin method, include the `--strict` option that is described in the following text.

--strict
Rebalances the file system with a strict round-robin method. In IBM Spectrum Scale 4.2.3 and earlier versions, rebalancing always uses this method.

Note: Rebalancing of files is an I/O intensive and time-consuming operation and is important only for file systems with large files that are mostly invariant. In many cases, normal file update and creation rebalance a file system over time without the cost of a complete rebalancing.

Note: Rebalancing distributes file blocks across all the disks in the cluster that are not suspended, including stopped disks. For stopped disks, rebalancing does not allow read operations and allocates data blocks without writing them to the disk. When the disk is restarted and replicated data is copied onto it, the file system completes the write operations.

-R
Changes the replication settings of each file, directory, and system metadata object so that they match the default file system settings (see the `mmchfs` command `-m` and `-r` options) as long as the maximum (`-M` and `-R`) settings for the object allow it. Next, it replicates or unreplicates the object as needed to match the new settings. This option can be used to replicate all of the existing files that were not previously replicated or to unreplicate the files if replication is no longer needed or wanted. All data is also migrated off disks that have either a `suspended` or `to be emptied` status.

--check-conflicting-replicas
Scans the file system and compares replicas of metadata and data for conflicts. Each such conflict is reported.

Attention: If this option reports inconsistent replicas, contact IBM Service for guidance.

-c [--read-only]
Deprecated in favor of the `--check-conflicting-replicas` option. Both the `-c` option and the `-c --read-only` option now have exactly same function as the `--check-conflicting-replicas` option. The function of `-c` in earlier releases, which was to attempt to fix replica conflicts, is no longer available. For more information, see the description of the option `--check-conflicting-replicas`.

--metadata-only
Limits the specified operation to metadata blocks. Data blocks are not affected. This option is valid only with the `-r`, `-b`, `-R`, or `--check-conflicting-replicas` option.

The `mmrestripefs` command with this option completes its operation quicker than a full restripe, replication, or replica compare of data and metadata.

Use this option when you want to prioritize the `mmrestripefs` operation on the metadata. This option ensures that the `mmrestripefs` operation has a reduced impact on the file system performance when compared to running the `mmrestripefs` command on the metadata and data.

After you run the `mmrestripefs` command on the metadata with *--metadata-only* option, you can issue the `mmrestripefs` command without this option to restripe the data and any metadata that requires to be restriped.

Note: This option does not run until all the nodes in the cluster are upgraded to IBM Spectrum Scale 4.2.1. If any of the nodes is not upgraded, the system displays the following error message:

```
mmrestripefs: The --metadata-only option support has not been enabled yet.
Issue "mmchconfig release=LATEST" to activate the new function.
mmrestripefs: Command failed. Examine previous error messages to determine cause.
```

-p
Directs `mmrestripefs` to repair the file placement within the storage pool.

Files that are assigned to one storage pool, but with data in a different pool, have their data migrated to the correct pool. Such files are referred to as ill-placed. Utilities, such as the `mmchattr` command, might change a file's storage pool assignment, but not move the data. The `mmrestripefs` command might then be invoked to migrate all of the data at once, rather than migrating each file individually. The placement option (`-p`) rebalances only the files that it moves. In contrast, the rebalance operation (`-b`) performs data placement on all files.

-z
Performs any deferred or incomplete file compression or decompression of files in the file system. For more information, see the topic *File compression* in the *IBM Spectrum Scale: Administration Guide*.

-P *PoolName*
Directs `mmrestripefs` to repair only the files that are assigned to the specified storage pool. This option is convenient for migrating ill-placed data blocks between pools, for example after you change a file's storage pool assignment with `mmchattr` or `mmapplypolicy` with the `-I defer` flag.

Do not use for other tasks, in particular, for any tasks that require metadata processing, such as re-replication. By design, all GPFS metadata is kept in the system pool, even for files that have blocks in other storage pools. Therefore, a command that must process all metadata must not be restricted to a specific storage pool.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specify the nodes that participate in the restripe of the file system. This command supports all defined node classes. The default is `all` or the current value of the `defaultHelperNodes` parameter of the `mmchconfig` command.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

-o *InodeResultFile*
Contains a list of the inodes that met the interesting inode flags that were specified on the `--inode-criteria` parameter. The output file contains the following:

INODE_NUMBER
This is the inode number.

DISKADDR
Specifies a dummy address for later `tsfindinode` use.

SNAPSHOT_ID
This is the snapshot ID.

ISGLOBAL_SNAPSHOT
Indicates whether or not the inode is in a global snapshot. Files in the live file system are considered to be in a global snapshot.

INDEPENDENT_FSETID
Indicates the independent fileset to which the inode belongs.

MEMO (*INODE_FLAGS FILE_TYPE* [*ERROR*])
Indicates the inode flag and file type that will be printed:

`Inode flags`:

```
BROKEN
exposed
dataUpdateMiss
illCompressed
illPlaced
illReplicated
metaUpdateMiss
unbalanced
```

`File types`:

```
BLK_DEV
CHAR_DEV
DIRECTORY
FIFO
LINK
LOGFILE
REGULAR_FILE
RESERVED
SOCK
*UNLINKED*
*DELETED*
```

Notes:

1. An error message will be printed in the output file if an error is encountered when repairing the inode.
2. `DISKADDR`, `ISGLOBAL_SNAPSHOT`, and `FSET_ID` work with the `tsfindinode` tool (`/usr/lpp/mmfs/bin/tsfindinode`) to find the file name for each inode. `tsfindinode` uses the output file to retrieve the file name for each interesting inode.

--inode-criteria *CriteriaFile*
Specifies the interesting inode criteria flag, where *CriteriaFile* contains a list of the following flags with one per line:

BROKEN
Indicates that a file has a data block with all of its replicas on disks that have been removed.

Note: `BROKEN` is always included in the list of flags even if it is not specified.

dataUpdateMiss
Indicates that at least one data block was not updated successfully on all replicas.

exposed
Indicates an inode with an exposed risk; that is, the file has data where all replicas are on suspended disks. This could cause data to be lost if the suspended disks have failed or been removed.

illCompressed
Indicates an inode in which file compression or decompression is deferred, or in which a compressed file is partly decompressed to allow the file to be written into or memory-mapped.

illPlaced
Indicates an inode with some data blocks that might be stored in an incorrect storage pool.

illReplicated
Indicates that the file has a data block that does not meet the setting for the replica.

metaUpdateMiss
Indicates that there is at least one metadata block that has not been successfully updated to all replicas.

unbalanced
Indicates that the file has a data block that is not well balanced across all the disks in all failure groups.

Note: If a file matches *any* of the specified interesting flags, all of its interesting flags (even those not specified) will be displayed.

--pit-continues-on-error
Allows the **mmrestripefs** command to continue repairing the remaining files, if errors are encountered in the parallel inode traverse (PIT) phase that performs scanning of user file metadata. An output file is generated if an error occurs in the PIT phase. The location of the file that logs the errors is displayed in the command output.

Note:

- The **mmrestripefs** command continues to run only if the error is reported in the PIT phase of the command execution. If the error is reported in other phases of command execution, the command stops running.
- The `--pit-continues-on-error` option works only if the minimum release level of the IBM Spectrum Scale cluster is 5.1.1 or later.

--qos ***QosClass***
Specifies the Quality of Service for I/O operations (QoS) class to which the instance of the command is assigned. If you do not specify this parameter, the instance of the command is assigned by default to the `maintenance` QoS class. This parameter has no effect unless the QoS service is enabled. For more information, see the topic "mmchqos command" on page 263. Specify one of the following QoS classes:

maintenance
This QoS class is typically configured to have a smaller share of file system IOPS. Use this class for I/O-intensive, potentially long-running GPFS commands, so that they contribute less to reducing overall file system performance.

other
This QoS class is typically configured to have a larger share of file system IOPS. Use this class for administration commands that are not I/O-intensive.

For more information, see the topic *Setting the Quality of Service for I/O operations (QoS)* in *IBM Spectrum Scale: Administration Guide*.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to issue the `mmrestripefs` command.

The node on which you issue the command must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To move all critical data from any suspended disk in file system `fs1`, issue the following command:

```
mmrestripefs fs1 -m
```

   The system displays information similar to the following output:

```
GPFS: 6027-589 Scanning file system metadata, phase 1 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 2 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 3 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 4 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-565 Scanning user file metadata ...
8.00 % complete on Tue Feb 24 16:56:55 2009 ( 708608 inodes 346 MB)
100.00 % complete on Tue Feb 24 16:56:56 2009
GPFS: 6027-552 Scan completed successfully.
```

2. To rebalance all files in file system `fs1` across all defined, accessible disks that are not stopped or suspended, issue the following command:

```
mmrestripefs fs1 -b
```

   The system displays information similar to the following output:

```
GPFS: 6027-589 Scanning file system metadata, phase 1 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 2 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 3 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 4 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-565 Scanning user file metadata ...
3.00 % complete on Tue Feb 24 16:56:39 2009 ( 180224 inodes 161 MB)
100.00 % complete on Tue Feb 24 16:56:44 2009
GPFS: 6027-552 Scan completed successfully.
```

3. The following command scans file system `gpfs1` for replica conflicts of metadata and data:

```
#mmrestripefs gpfs1 --check-conflicting-replicas
Scanning file system metadata, phase 1 ...
Inode 0 in fileset 0 and snapshot 0 has mismatch in replicated disk address 2:104859136
Scan completed successfully.
Scanning file system metadata, phase 2 ...
Scan completed successfully.
Scanning file system metadata, phase 3 ...
Scan completed successfully.
Scanning file system metadata, phase 4 ...
Scan completed successfully.
Scanning user file metadata ...
 100.00 % complete on Tue Jul 30 03:32:44 2013
Scan completed successfully.
```

4. To fix the pool placement of files in file system `fs1` and also determine which files are illReplicated (for example, as a result of a failed disk), issue the following command:

```
mmrestripefs fs1 -p --inode-criteria /tmp/crit -o /tmp/inodeResultFile
```

   The system displays information similar to the following output:

```
GPFS: 6027-589 Scanning file system metadata, phase 1 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 2 ...
Scanning file system metadata for data storage pool
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 3 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 4 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-565 Scanning user file metadata ...
```

```
 100.00 % complete on Wed Apr 15 10:15:15 2015  (65792 inodes with total  400 MB data
processed)
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-3902 Check file '/tmp/inodeResultFile' on vmip1 for inodes that were \
                          found matching the criteria.
#10:15:15# vmip1:/fs1 # cat /tmp/crit
illReplicated
#10:15:19# vmip1:/fs1 # cat /tmp/inodeResultFile
This inode list was generated in the Parallel Inode Traverse on Wed Apr 15 10:15:14 2015
INODE_NUMBER DISKADDR SNAPSHOT_ID ISGLOBAL_SNAPSHOT FSET_ID MEMO(INODE_FLAGS FILE_TYPE
[ERROR])
 24320        0:0      0           1                 0       illreplicated unbalanced
REGULAR_FILE
 24322        0:0      0           1                 0       illreplicated unbalanced
REGULAR_FILE
 24321        0:0      0           1                 0       illreplicated unbalanced
REGULAR_FILE
 24324        0:0      0           1                 0       illreplicated unbalanced
REGULAR_FILE
 24325        0:0      0           1                 0       illreplicated unbalanced
REGULAR_FILE
 24323        0:0      0           1                 0       illreplicated unbalanced
REGULAR_FILE
 24326        0:0      0           1                 0       illreplicated unbalanced
REGULAR_FILE
 24327        0:0      0           1                 0       illreplicated unbalanced
REGULAR_FILE
 24328        0:0      0           1                 0       illreplicated unbalanced
REGULAR_FILE
 24329        0:0      0           1                 0       illreplicated unbalanced
REGULAR_FILE
```

5. The following command moves all critical data from any suspended disk in the file system `gpfs0`, and also determines which files with data are failed to migrate (for example, as a result of a failed disk I/O):

   ```
   # mmrestripefs gpfs0 -m
   ```

   The system displays information similar to the following output:

   ```
   GPFS: 6027-589 Scanning file system metadata, phase 1 ...
      GPFS: 6027-552 Scan completed successfully.
      GPFS: 6027-589 Scanning file system metadata, phase 2 ...
      GPFS: 6027-552 Scan completed successfully.
      GPFS: 6027-589 Scanning file system metadata, phase 3 ...
      GPFS: 6027-552 Scan completed successfully.
      GPFS: 6027-589 Scanning file system metadata, phase 4 ...
      GPFS: 6027-552 Scan completed successfully.
      GPFS: 6027-565 Scanning user file metadata ...
       83.11 % complete on Mon Feb 22 02:45:04 2021  (    200640 inodes with total       7423
   MB data processed)
      100.00 % complete on Mon Feb 22 02:45:24 2021  (    501760 inodes with total      10858
   MB data processed)
      GPFS: 6027-566 Error processing user file metadata.
      No space left on device
   Check the file /var/mmfs/tmp/gpfs0.pit.interestingInodes.8589934594 on scale-002 for inodes
   with broken disk addresses or failures.
   mmrestripefs: tsrestripefs failed.
   ```

See also

- “mmadddisk command” on page 28
- “mmapplypolicy command” on page 80
- “mmchattr command” on page 157
- “mmchdisk command” on page 212
- “mmchfs command” on page 232
- “mmdeldisk command” on page 364
- “mmrpldisk command” on page 690
- “mmrestripefile command” on page 678

Location

/usr/lpp/mmfs/bin

mmrpldisk command

Replaces the specified disk.

Synopsis

```
mmrpldisk Device DiskName {DiskDesc | -F StanzaFile} [-v {yes | no}]
                [-N {Node[,Node...] | NodeFile | NodeClass}]
                [--inode-criteria CriteriaFile] [-o InodeResultFile]
                [--qos QOSClass]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmrpldisk` command to replace an existing disk in the GPFS file system with a new one. All data on the old disk is migrated to the new disk.

To replace a disk in a GPFS file system, you must first decide if you will:

1. Create a new disk using the `mmcrnsd` command.

   In this case, use the rewritten disk stanza file produced by the `mmcrnsd` command or create a new disk stanza. When using the rewritten file, the disk usage and failure group specifications remain the same as specified on the `mmcrnsd` command.
2. Select a disk no longer in any file system. Issue the `mmlsnsd -F` command to display the available disks.

The disk may then be used to replace a disk in the file system using the `mmrpldisk` command.

Notes:

- Do not replace a stopped disk under any circumstances. You must start the disk before replacing it. If the disk cannot be started, delete it with the `mmdeldisk` command. For more information, see the topic *Disk media failure* in the *IBM Spectrum Scale: Problem Determination Guide*.
- A disk cannot be replaced if it is the only disk in the file system.
- The replacement disk must have the same thin disk type as the disk being replaced. Otherwise the `mmrpldisk` command fails with an error message. For more information, see the description of the `thinDiskType` parameter later in this help topic.
- The `mmrpldisk` command can be run while a file system is mounted.

Results

Upon successful completion of the `mmrpldisk` command, the disk is replaced in the file system and data is copied to the new disk without restriping.

Parameters

Device
The device name of the file system where the disk is to be replaced. File system names need not be fully-qualified. `fs0` is as acceptable as `/dev/fs0`.

This must be the first parameter.

DiskName
The name of the disk to be replaced. To display the names of disks that belong to the file system, issue the `mmlsnsd -f`, `mmlsfs -d`, or `mmlsdisk` command. The `mmlsdisk` command will also show the current disk usage and failure group values for each of the disks.

DiskDesc
A descriptor for the replacement disk.

Prior to GPFS 3.5, the disk information for the `mmrpldisk` command was specified in the form of a disk descriptor defined as follows (with the second, third, sixth, and seventh fields reserved):

```
DiskName:::DiskUsage:FailureGroup:::
```

For backward compatibility, the `mmrpldisk` command will still accept a traditional disk descriptor as input, but this use is discouraged.

-F *StanzaFile*
Specifies a file containing the NSD stanzas for the replacement disk. NSD stanzas have this format:

```
%nsd:
  nsd=NsdName
  usage={dataOnly | metadataOnly | dataAndMetadata | descOnly}
  failureGroup=FailureGroup
  pool=StoragePool
  servers=ServerList
  device=DiskName
  thinDiskType={no | nvme | scsi | auto}
```

where:

nsd=*NsdName*
The name of an NSD previously created by the `mmcrnsd` command. For a list of available disks, issue the `mmlsnsd -F` command. This clause is mandatory for the `mmrpldisk` command.

usage={dataOnly | metadataOnly | dataAndMetadata | descOnly}
Specifies the type of data to be stored on the disk:

dataAndMetadata
Indicates that the disk contains both data and metadata. This is the default for disks in the system pool.

dataOnly
Indicates that the disk contains data and does not contain metadata. This is the default for disks in storage pools other than the system pool.

metadataOnly
Indicates that the disk contains metadata and does not contain data.

descOnly
Indicates that the disk contains no data and no file metadata. IBM Spectrum Scale uses this type of disk primarily to keep a copy of the file system descriptor. It can also be used as a third failure group in certain disaster recovery configurations. For more information, see the topic *Synchronous mirroring utilizing GPFS replication* in the *IBM Spectrum Scale: Administration Guide*.

This clause is optional for the `mmrpldisk` command. If omitted, the new disk will inherit the usage type of the disk being replaced.

failureGroup=*FailureGroup*
Identifies the failure group to which the disk belongs. A failure group identifier can be a simple integer or a topology vector that consists of up to three comma-separated integers. The default is -1, which indicates that the disk has no point of failure in common with any other disk.

GPFS uses this information during data and metadata placement to ensure that no two replicas of the same block can become unavailable due to a single failure. All disks that are attached to the same NSD server or adapter must be placed in the same failure group.

If the file system is configured with data replication, all storage pools must have two failure groups to maintain proper protection of the data. Similarly, if metadata replication is in effect, the system storage pool must have two failure groups.

Disks that belong to storage pools in which write affinity is enabled can use topology vectors to identify failure domains in a shared-nothing cluster. Disks that belong to traditional storage pools must use simple integers to specify the failure group.

This clause is optional for the `mmrpldisk` command. If omitted, the new disk will inherit the failure group of the disk being replaced.

pool=*StoragePool*
Specifies the storage pool to which the disk is to be assigned. This clause is ignored by the `mmrpldisk` command.

servers=*ServerList*
A comma-separated list of NSD server nodes. This clause is ignored by the `mmrpldisk` command.

device=*DiskName*
The block device name of the underlying disk device. This clause is ignored by the `mmrpldisk` command.

thinDiskType={no | nvme | scsi | auto}
Specifies the space reclaim disk type:

Attention: The replacement disk must have the thin disk type as the disk being replaced. Otherwise the `mmrpldisk` command fails with an error message. For more information, see the topic *IBM Spectrum Scale with data reduction storage devices* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

no
The disk does not support space reclaim. This value is the default.

nvme
The disk is a TRIM capable NVMe device that supports the `mmreclaimspace` command.

scsi
The disk is a thin provisioned SCSI disk that supports the `mmreclaimspace` command.

auto
The type of the disk is either `nvme` or `scsi`. IBM Spectrum Scale will try to detect the actual disk type automatically. To avoid problems, you should replace `auto` with the correct disk type, `nvme` or `scsi`, as soon as you can.

Note: In 5.0.5, the space reclaim auto-detection is enhanced. It is encouraged to use the auto key-word after your cluster is upgraded to 5.0.5.

For more information, see the topic *IBM Spectrum Scale with data reduction storage devices* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Note: While it is not absolutely necessary to specify the same parameters for the new disk as the old disk, it is suggested that you do so. If the new disk is equivalent in size to the old disk, and if the disk usage and failure group parameters are the same, the data and metadata can be completely migrated from the old disk to the new disk. A disk replacement in this manner allows the file system to maintain its current data and metadata balance.

If the new disk has a different size, disk usage, parameter, or failure group parameter, the operation may leave the file system unbalanced and require a restripe. Additionally, a change in size or the disk usage parameter may cause the operation to fail since other disks in the file system may not have sufficient space to absorb more data or metadata. In this case, first use the `mmadddisk` command to add the new disk, the `mmdeldisk` command to delete the old disk, and finally the `mmrestripefs` command to rebalance the file system.

-v {yes | no}
Verify the new disk does not belong to an existing file system. The default is `-v yes`. Specify `-v no` only when you want to reuse a disk that is no longer needed for an existing file system. If the command is interrupted for any reason, use the `-v no` option on the next invocation of the command.

Important: Using `-v no` on a disk that already belongs to a file system will corrupt that file system. This will not be noticed until the next time that file system is mounted.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specify the nodes that participate in the migration of data from the old to the new disk. This command supports all defined node classes. The default is `all` or the current value of the `defaultHelperNodes` parameter of the `mmchconfig` command.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

--inode-criteria *CriteriaFile*
Specifies the interesting inode criteria flag, where *CriteriaFile* contains a list of the following flags with one per line:

BROKEN
Indicates that a file has a data block with all of its replicas on disks that have been removed.

Note: BROKEN is always included in the list of flags even if it is not specified.

dataUpdateMiss
Indicates that at least one data block was not updated successfully on all replicas.

exposed
Indicates an inode with an exposed risk; that is, the file has data where all replicas are on suspended disks. This could cause data to be lost if the suspended disks have failed or been removed.

illCompressed
Indicates an inode in which file compression or decompression is deferred, or in which a compressed file is partly decompressed to allow the file to be written into or memory-mapped.

illPlaced
Indicates an inode with some data blocks that might be stored in an incorrect storage pool.

illReplicated
Indicates that the file has a data block that does not meet the setting for the replica.

metaUpdateMiss
Indicates that there is at least one metadata block that has not been successfully updated to all replicas.

unbalanced
Indicates that the file has a data block that is not well balanced across all the disks in all failure groups.

Note: If a file matches *any* of the specified interesting flags, all of its interesting flags (even those not specified) will be displayed.

-o *InodeResultFile*
Contains a list of the inodes that met the interesting inode flags that were specified on the `--inode-criteria` parameter. The output file contains the following:

INODE_NUMBER
This is the inode number.

DISKADDR
Specifies a dummy address for later `tsfindinode` use.

SNAPSHOT_ID
This is the snapshot ID.

ISGLOBAL_SNAPSHOT
Indicates whether or not the inode is in a global snapshot. Files in the live file system are considered to be in a global snapshot.

INDEPENDENT_FSETID
Indicates the independent fileset to which the inode belongs.

MEMO (*INODE_FLAGS FILE_TYPE* [*ERROR*])
Indicates the inode flag and file type that will be printed:

Inode flags:

```
BROKEN
exposed
dataUpdateMiss
illCompressed
illPlaced
illReplicated
metaUpdateMiss
unbalanced
```

File types:

```
BLK_DEV
CHAR_DEV
DIRECTORY
FIFO
LINK
LOGFILE
REGULAR_FILE
RESERVED
SOCK
*UNLINKED*
*DELETED*
```

Notes:

1. An error message will be printed in the output file if an error is encountered when repairing the inode.
2. DISKADDR, ISGLOBAL_SNAPSHOT, and FSET_ID work with the `tsfindinode` tool (`/usr/lpp/mmfs/bin/tsfindinode`) to find the file name for each inode. `tsfindinode` uses the output file to retrieve the file name for each interesting inode.

--qos *QosClass*
Specifies the Quality of Service for I/O operations (QoS) class to which the instance of the command is assigned. If you do not specify this parameter, the instance of the command is assigned by default to the `maintenance` QoS class. This parameter has no effect unless the QoS service is enabled. For more information, see the topic "mmchqos command" on page 263. Specify one of the following QoS classes:

maintenance
This QoS class is typically configured to have a smaller share of file system IOPS. Use this class for I/O-intensive, potentially long-running GPFS commands, so that they contribute less to reducing overall file system performance.

other
This QoS class is typically configured to have a larger share of file system IOPS. Use this class for administration commands that are not I/O-intensive.

For more information, see the topic *Setting the Quality of Service for I/O operations (QoS)* in *IBM Spectrum Scale: Administration Guide*.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmrpldisk` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To replace disk hd27n01 in fs1 with a new disk, hd16vsdn10 allowing the disk usage and failure group parameters to default to the corresponding values of hd27n01, and have only nodes **c154n01**, **c154n02**, and **c154n09** participate in the migration of the data, issue the following command:

```
# mmrpldisk fs1 hd27n01 hd16vsdn10 -N c154n01,c154n02,c154n09
```

A sample output is as follows:

```
Replacing hd27n01 ...

The following disks of fs1 will be formatted on node c155n01.ppd.pok.ibm.com:
    hd16vsdn10: size 17793024 KB
Extending Allocation Map
Checking Allocation Map for storage pool 'system'
   7 % complete on Wed May 16 16:36:30 2007
  18 % complete on Wed May 16 16:36:35 2007
  34 % complete on Wed May 16 16:36:40 2007
  49 % complete on Wed May 16 16:36:45 2007
  65 % complete on Wed May 16 16:36:50 2007
  82 % complete on Wed May 16 16:36:55 2007
  98 % complete on Wed May 16 16:37:00 2007
 100 % complete on Wed May 16 16:37:01 2007
Completed adding disks to file system fs1.
Scanning system storage pool
Scanning file system metadata, phase 1 ...
   2 % complete on Wed May 16 16:37:04 2007
   7 % complete on Wed May 16 16:37:11 2007
  14 % complete on Wed May 16 16:37:18 2007
  20 % complete on Wed May 16 16:37:24 2007
  27 % complete on Wed May 16 16:37:31 2007
  34 % complete on Wed May 16 16:37:37 2007
  50 % complete on Wed May 16 16:37:50 2007
  61 % complete on Wed May 16 16:38:00 2007
  68 % complete on Wed May 16 16:38:06 2007
  79 % complete on Wed May 16 16:38:16 2007
  90 % complete on Wed May 16 16:38:26 2007
 100 % complete on Wed May 16 16:38:32 2007
Scan completed successfully.
Scanning file system metadata, phase 2 ...
Scanning file system metadata for fs1sp1 storage pool
Scan completed successfully.
Scanning file system metadata, phase 3 ...
Scan completed successfully.
Scanning file system metadata, phase 4 ...
Scan completed successfully.
Scanning user file metadata ...
   3 %  complete on Wed May 16 16:38:38 2007
  25 %  complete on Wed May 16 16:38:47 2007
  53 %  complete on Wed May 16 16:38:53 2007
  87 %  complete on Wed May 16 16:38:59 2007
  97 %  complete on Wed May 16 16:39:06 2007
 100 %  complete on Wed May 16 16:39:07 2007
Scan completed successfully.
Done
mmrpldisk: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

2. To replace disk vmip3_nsd1 from storage pool GOLD on file system fs2 and to search for any interesting files handled during the mmrpldisk at the same time, issue the following command:

```
# mmrpldisk fs2 vmip3_nsd1 -F f /tmp/crit --inode-criteria
```

A sample output is as follows:

```
Replacing vmip3_nsd1 ...

GPFS: 6027-531 The following disks of fs2 will be formatted on node vmip1:
    vmip2_nsd3: size 5120 MB
Extending Allocation Map
Checking Allocation Map for storage pool GOLD
  59 % complete on Wed Apr 15 10:52:44 2015
```

```
 100 % complete on Wed Apr 15 10:52:49 2015
GPFS: 6027-1503 Completed adding disks to file system fs2.
GPFS: 6027-589 Scanning file system metadata, phase 1 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 2 ...
Scanning file system metadata for GOLD storage pool
Scanning file system metadata for BRONZE storage pool
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 3 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 4 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-565 Scanning user file metadata ...
   6.47 % complete on Wed Apr 15 10:53:11 2015  (     65792 inodes with total        448 MB data
processed)
   6.49 % complete on Wed Apr 15 10:55:01 2015  (     65792 inodes with total        448 MB data
processed)
 100.00 % complete on Wed Apr 15 10:55:03 2015  (     65792 inodes with total        448 MB data
processed)
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-3902 Check file '/var/mmfs/tmp/fs2.pit.interestingInodes.12884901928' on vmip1 for inodes \
                          that were found matching the criteria.
Checking Allocation Map for storage pool GOLD
  56 % complete on Wed Apr 15 10:55:08 2015
 100 % complete on Wed Apr 15 10:55:12 2015
Done
mmrpldisk: 6027-1371 Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
#11:57:08# vmip1:/fs2 # cat /tmp/crit
illReplicated
illPlaced
dataUpdateMiss
metaUpdateMiss
exposed
BROKEN
#11:09:24# vmip1:/fs2 # cat /var/mmfs/tmp/fs2.pit.interestingInodes.12884901928
This inode list was generated in the Parallel Inode Traverse on Wed Apr 15 10:55:02 2015
INODE_NUMBER DISKADDR SNAPSHOT_ID ISGLOBAL_SNAPSHOT FSET_ID MEMO(INODE_FLAGS FILE_TYPE [ERROR])
 50177        0:0      0           1                 0       illplaced REGULAR_FILE
```

Note: The `mmrpldisk` command will report any interesting inodes that it finds during routine processing, but the list might not be 100% accurate or complete.

See also

- “mmadddisk command” on page 28
- “mmchdisk command” on page 212
- “mmcrnsd command” on page 335
- “mmlsdisk command” on page 497
- “mmlsnsd command” on page 522
- “mmrestripefs command” on page 682

Location

`/usr/lpp/mmfs/bin`

mmsdrrestore command

Restores the latest GPFS system files on the specified nodes.

Synopsis

```
mmsdrrestore [-p NodeName] [-F mmsdrfsFile] [-R remoteFileCopyCommand]
             [-a | -N {Node[,Node...] | NodeFile | NodeClass}]
```

or

```
mmsdrrestore --ccr-repair [-p NodeName] [-F mmsdrfsFile] [-R remoteFileCopyCommand]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmsdrrestore` command is intended for use by experienced system administrators.

The `mmsdrrestore` command restores the latest GPFS system files on the specified nodes. If no nodes are specified, the command restores the configuration information only on the node on which the command is issued. If the local GPFS configuration file is missing, the file that is specified with the `-F` option from the node that is specified with the `-p` option is used instead.

This command works best when the `-F` option specifies a backup file that is created by the `mmsdrbackup` user exit. If the Cluster Configuration Repository (CCR) is enabled, the `mmsdrbackup` user exit creates a CCR backup file. If the CCR is not enabled, the user exit creates an `mmsdrfs` backup file. For more information, see “mmsdrbackup user exit” on page 1010.

The `mmsdrrestore` command cannot restore a cluster configuration unless a majority of the quorum nodes in the cluster are accessible. However, this requirement does not apply if the `-F` option specifies a CCR backup file or if the `--ccr-repair` option is specified.

Parameters

-p *NodeName*
Specifies the node from which to obtain a valid GPFS configuration file. The node must be either the primary configuration server or a node that has a valid backup copy of the `mmsdrfs` file. If this parameter is not specified, the command uses the configuration file on the node from which the command is issued.

-F *mmsdrfsFile*
Specifies the path name of the GPFS configuration file for the `mmsdrrestore` command to use. This configuration file might be the current one on the primary server, or it might be a configuration file that is obtained from the `mmsdrbackup` user exit. If not specified, `/var/mmfs/gen/mmsdrfs` is used.

If the configuration file is a CCR backup file, you must also specify the `-a` option of the `mmsdrrestore` command. The command restores any nodes in the cluster that need to be restored. If the configuration file is an `mmsdrfs` file, you can specify the nodes to be restored with the `-N` option or you can issue the command from a node that needs to be restored.

-R *remoteFileCopyCommand*
Specifies the fully qualified path name for the remote file copy program to be used for obtaining the GPFS configuration file. The default is `/usr/bin/rcp`.

-a
Restores the GPFS configuration files on all nodes in the cluster.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Restores the GPFS configuration files on a set of nodes.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

This option does not support a *NodeClass* of `mount`.

In IBM Spectrum Scale 5.0.4 and later, the `-N` option is valid not only when the cluster is using server-based backup but also when the cluster has the CCR enabled.

`--ccr-repair`
Repairs corrupted or lost files in the CCR `committed` directory of all of the quorum nodes. Use this option only when the CCR `committed` directories of all of the quorum nodes have corrupted or lost files and other restore methods have failed. For more information, see Example 4 and *Repair of cluster configuration information when no CCR backup information is available:* ***`mmsdrrestore` command*** in the *IBM Spectrum Scale: Problem Determination Guide*.

Be aware of the following considerations when you issue the `mmsdrrestore` command with the `--ccr-repair` option:

- The command runs only if all the quorum nodes have corrupted or lost files in the CCR `committed` directory.
- You must shut down the GPFS daemon on all the quorum nodes before you run the command.

In IBM Spectrum Scale 5.0.4 and later you can use the `--ccr-repair` option in the following environments:

- On nodes that are running Linux, AIX, or Microsoft Windows
- In a sudo-wrapper environment on Linux or AIX

The `--ccr-repair` option requires a version of Python to be installed on the node:

- If the node is running IBM Spectrum Scale 5.1.0 or later, Python 3 or later must be installed on the node:
  - On Linux, Python 3 is usually installed automatically.
  - On AIX, manually install Python 3.0 or later from the AIX Toolbox for Linux Applications.
  - On Windows, manually install Python 3 under the Cygwin environment as described in the following steps. Windows native (non-Cygwin) distributions of Python 3 are not supported.
    1. From http://www.cygwin.com, download and run the Cygwin 64-bit setup program `setup-x86_64.exe`.
    2. In the "Select Packages" window, click **View** > **Category**.
    3. Click **All** > **Python** > **Python3** and select the latest level.
    4. Follow the instructions to complete the installation.
- If the node is running IBM Spectrum Scale 5.0.5.x, Python 2 must be installed on the node:
  - On Linux, Python 2 is usually installed automatically.
  - On AIX, manually install Python 2.7.5 or a later version of Python 2 from the AIX Toolbox for Linux Applications.
  - On Windows, manually install Python 2 under the Cygwin environment as described in the following steps. Windows native (non-Cygwin) distributions of Python 2 are not supported.
    1. From http://www.cygwin.com, download and run the Cygwin 64-bit setup program `setup-x86_64.exe`.
    2. In the "Select Packages" window, click **View** > **Category**.
    3. Click **All** > **Python** > **Python2** and select the latest level.
    4. Follow the instructions to complete the installation.

Important: The use of the `--ccr-repair` option does not guarantee the recovery of the most recent state of all the configuration files in the CCR. Instead, the option brings the CCR back into a consistent state with the most recent available version of each configuration file.

Exit status

0
Successful completion.

nonzero
A failure occurred.

Security

You must have root authority to run the `mmsdrrestore` command.

The node on which the command is issued must be able to run remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To restore the latest GPFS system files on the local node using the GPFS configuration file `/var/mmfs/gen/mmsdrfs` from the node that is named `primaryServer`, issue the following command:

```
# mmsdrrestore -p primaryServer
```

   A sample output is as follows:

```
Tue Jul  3 18:19:53 CDT 2012: mmsdrrestore: Processing node  k164n04.kgn.ibm.com
mmsdrrestore:  Node k164n04.kgn.ibm.com successfully restored.
```

2. To restore the GPFS system files on all nodes in the cluster using GPFS configuration file `/GPFSconfigFiles/mmsdrfs.120605` on the node that is named `GPFSarchive`, issue the following command from the node named `localNode`:

```
# mmsdrrestore -p GPFSarchive -F /GPFSconfigFiles/mmsdrfs.120605 -a
```

   A sample output is as follows:

```
Tue Jul  3 18:29:28 CDT 2012: mmsdrrestore: Processing node k164n04.kgn.ibm.com
Tue Jul  3 18:29:30 CDT 2012: mmsdrrestore: Processing node k164n05.kgn.ibm.com
Tue Jul  3 18:29:31 CDT 2012: mmsdrrestore: Processing node k164n06.kgn.ibm.com
mmsdrrestore: Command successfully completed
```

3. The following command restores the GPFS system files from a CCR backup file. The `-a` option is required. The command restores any nodes in the cluster that need to be restored:

```
# mmsdrrestore -F /GPFSbackupFiles/CCRBackup.2015.10.14.10.01.25.tar.gz -a
```

   A sample output is as follows:

```
Restoring CCR backup
CCR backup has been restored
```

4. The following example shows how the `mmsdrrestore` command can be run with the `--ccr-repair` option to repair missing or corrupted files in the CCR `committed` directory of all of the quorum nodes.

   Note: The command returns with an error message if the CCR `committed` directory of any of the quorum nodes does not have corrupted or lost files.

a. Issue a command like the following one to shut down the GPFS daemon on all of the quorum nodes:

```
# mmshutdown -a
Wed Mar 13 01:57:39 EDT 2019: mmshutdown: Starting force unmount of GPFS file systems
Wed Mar 13 01:57:44 EDT 2019: mmshutdown: Shutting down GPFS daemons
Wed Mar 13 01:58:07 EDT 2019: mmshutdown: Finished
```

b. Issue the following command to repair the missing or corrupted CCR files:

```
# mmsdrrestore --ccr-repair
mmsdrrestore: Checking CCR on all quorum nodes ...
mmsdrrestore. Invoking CCR restore in dry run mode ...

ccrrestore: +++ DRY RUN: CCR state on quorum nodes and tiebreaker disks will not be
restored +++
ccrrestore:  1/10: Test tool chain successful
ccrrestore:  2/10: Setup local working directories successful
ccrrestore:  3/10: Read CCR Paxos state from tiebreaker disks successful
ccrrestore:  4/10: Copy Paxos state files from quorum nodes successful
ccrrestore:  5/10: Getting most recent Paxos state file successful
ccrrestore:  6/10: Get cksum of files in committed directory successful
ccrrestore:  7/10: WARNING: Intact ccr.nodes file missing in committed directory
ccrrestore:  7/10: INFORMATION: Intact mmsysmon.json found (file id: 3 version: 1)
ccrrestore:  7/10: INFORMATION: Intact mmsdrfs found (file id: 4 version: 901)
ccrrestore:  7/10: INFORMATION: Intact mmLockFileDB found (file id: 5 version: 1)
ccrrestore:  7/10: INFORMATION: Intact genKeyData found (file id: 6 version: 1)
ccrrestore:  7/10: INFORMATION: Intact genKeyDataNew found (file id: 7 version: 1)
ccrrestore:  7/10: Parsing committed file list successful
ccrrestore:  8/10: Get cksum of CCR files successful
ccrrestore:  9/10: Pulling committed files from quorum nodes successful
ccrrestore: 10/10: File name: 'ccr.nodes' file state: UPDATED remark: 'OLD (v1, ((n1,e1),
0),
     9a5d4266)'
ccrrestore: 10/10: File name: 'ccr.disks' file state: MATCHING remark: 'none'
ccrrestore: 10/10: File name: 'mmsysmon.json' file state: MATCHING remark: 'none'
ccrrestore: 10/10: File name: 'mmsdrfs' file state: MATCHING remark: 'none'
ccrrestore: 10/10: File name: 'mmLockFileDB' file state: MATCHING remark: 'none'
ccrrestore: 10/10: File name: 'genKeyData' file state: MATCHING remark: 'none'
ccrrestore: 10/10: File name: 'genKeyDataNew' file state: MATCHING remark: 'none'
ccrrestore: 10/10: Patching Paxos state successful

mmsdrrestore: Review the dry run report above to see what will be changed and decide if
you
     want to continue the restore or not.  Do you want to continue? (yes/no) yes
ccrrestore:  1/17: Test tool chain successful
ccrrestore:  2/17: Test GPFS shutdown successful
ccrrestore:  3/17: Setup local working directories successful
ccrrestore:  4/17: Archiving CCR directories on quorum nodes successful
ccrrestore:  5/17: Read CCR Paxos state from tiebreaker disks successful
ccrrestore:  6/17: Kill GPFS mmsdrserv daemon successful
ccrrestore:  7/17: Copy Paxos state files from quorum nodes successful
ccrrestore:  8/17: Getting most recent Paxos state file successful
ccrrestore:  9/17: Get cksum of files in committed directory successful
ccrrestore: 10/17: WARNING: Intact ccr.nodes file missing in committed directory
ccrrestore: 10/17: INFORMATION: Intact mmsysmon.json found (file id: 3 version: 1)
ccrrestore: 10/17: INFORMATION: Intact mmsdrfs found (file id: 4 version: 901)
ccrrestore: 10/17: INFORMATION: Intact mmLockFileDB found (file id: 5 version: 1)
ccrrestore: 10/17: INFORMATION: Intact genKeyData found (file id: 6 version: 1)
ccrrestore: 10/17: INFORMATION: Intact genKeyDataNew found (file id: 7 version: 1)
ccrrestore: 10/17: Parsing committed file list successful
ccrrestore: 11/17: Get cksum of CCR files successful
ccrrestore: 12/17: Pulling committed files from quorum nodes successful
ccrrestore: 13/17: File name: 'ccr.nodes' file state: UPDATED remark: 'OLD (v1, ((n1,e1),
0),
     9a5d4266)'
ccrrestore: 13/17: File name: 'ccr.disks' file state: MATCHING remark: 'none'
ccrrestore: 13/17: File name: 'mmsysmon.json' file state: MATCHING remark: 'none'
ccrrestore: 13/17: File name: 'mmsdrfs' file state: MATCHING remark: 'none'
ccrrestore: 13/17: File name: 'mmLockFileDB' file state: MATCHING remark: 'none'
ccrrestore: 13/17: File name: 'genKeyData' file state: MATCHING remark: 'none'
ccrrestore: 13/17: File name: 'genKeyDataNew' file state: MATCHING remark: 'none'
ccrrestore: 13/17: Patching Paxos state successful
ccrrestore: 14/17: Pushing CCR files successful
ccrrestore: 15/17: Started GPFS mmsdrserv daemon successful
ccrrestore: 16/17: Ping GPFS mmsdrserv daemon successful
ccrrestore: 17/17: Write CCR Paxos state to tiebreaker disks
```

c. Issue a command like the following one to restart the GPFS daemon on all the quorum nodes:

```
# mmstartup -a
Tue Mar 19 22:31:35 EDT 2019: mmstartup: Starting GPFS ...
```

See also

- “mmsdrbackup user exit” on page 1010

Location

/usr/lpp/mmfs/bin

mmsetquota command

Sets quota limits.

Synopsis

```
mmsetquota Device{[:FilesetName]
            [--user IdOrName[,IdOrName]] [--group IdOrName[,IdOrName]]}
           {[--block SoftLimit[:HardLimit]] [--files SoftLimit[:HardLimit]]}
```

or

```
mmsetquota Device[:FilesetName] --default {user | group}
           {[--block SoftLimit[:HardLimit]] [--files SoftLimit[:HardLimit]]}
```

or

```
mmsetquota Device --default fileset
           {[--block SoftLimit[:HardLimit]] [--files SoftLimit[:HardLimit]]}
```

or

```
mmsetquota Device --grace {user | group | fileset}
           {[--block GracePeriod] [--files GracePeriod]}
```

or

```
mmsetquota -F StanzaFile
```

Availability

Available on all IBM Spectrum Scale editions. Available on AIX and Linux.

Description

The `mmsetquota` command sets quota limits, default quota limits, or grace periods for users, groups, and file sets in the specified file system.

When setting quota limits for a file system, replication within the file system should be considered. For explanation, see *Listing quotas* in *IBM Spectrum Scale: Administration Guide*

Important: Quota limits are not enforced for root users (by default). For information on managing quotas, see *Managing GPFS quotas* in the *IBM Spectrum Scale: Administration Guide*.

Parameters

Device
Specifies the device name of the file system.

FilesetName
Specifies the name of a fileset located on *Device* for which quota information is to be set.

IdOrName
Specifies a numeric ID, user name, or group name.

SoftLimit
Specifies the amount of data or the number of files the user, group, or fileset will be allowed to use.

HardLimit
Specifies the amount of data or the number of files the user, group, or fileset will be allowed to use during a grace period. If omitted, the default is no limit. See note.

GracePeriod
Specifies the file-system grace period during which quotas can exceed the soft limit before it is imposed as a hard limit. See note.

StanzaFile
Specifies a file containing quota stanzas.

--block
Specifies the quota limits or grace period for data.

--files
Specifies the quota limits or grace period for files.

--default
Sets the default quota for the user, group, or fileset.

--grace
Sets the grace period for the user, group, or fileset.

-F *StanzaFile*
Specifies a file containing the quota stanzas for set quota, set default quota, or set grace period.

Quota stanzas have this format:

```
%quota:
   device=Device
   command={setQuota|setDefaultQuota|setGracePeriod}
   type={USR|GRP|FILESET}
   id=IdList
   fileset=FilesetName
   blockQuota=Number
   blockLimit=Number
   blockGrace=Period
   filesQuota=Number
   filesLimit=Number
   filesGrace=Period
```

where:

device=*Device*
The device name of file system.

command={setQuota|setDefaultQuota|setGracePeriod}
Specifies the command to be executed for this stanza.

setQuota
Sets the quota limits. This command ignores `blockGrace` and `filesGrace` attributes.

setDefaultQuota
Sets the default quota limits. This command ignores `id`, `blockGrace` and `filesGrace` attributes.

setGracePeriod
Sets the grace periods. The command ignores `id`, `fileset`, and quota limit attributes. Grace periods can be set for each quota type in the file system.

type={USR|GRP|FILESET}
Specifies whether the command applies to user, group, or fileset.

id=*IdList*
Specifies a list of numeric IDs or user, group, or fileset names.

fileset=*FilesetName*
Specifies the fileset name for the `perfileset` quota setting. This attribute is ignored for `type=FILESET`

blockQuota=*Number*
Specifies the block soft limit. The number can be specified using the suffix K, M, G, or T. See note.

blockLimit=*Number*
Specifies the block hard limit. The number can be specified using the suffix K, M, G, or T. See note.

filesQuota=*Number*
Specifies the inode soft limit. The number can be specified using the suffix K, M, or G. See note.

filesLimit=*Number*
Specifies the inode hard limit. The number can be specified using the suffix K, M, or G. See note.

blockGrace=*Period*
Specifies the file-system grace period during which the block quotas can exceed the soft limit before it is imposed as a hard limit. The period can be specified in days, hours, minutes, or seconds.

filesGrace=*Period*
Specifies the file-system grace period during which the files quota can exceed the soft limit before it is imposed as a hard limit. The period can be specified in days, hours, minutes, or seconds.

Note:

- The maximum files limit is 2147483647.
- The maximum block limit is 999999999999999K. For values greater than 976031318016K (909T) and up to the maximum limit of 999999999999999K (about 931322T), you must specify the equivalent value with the suffix K, M, or G.
- If you want to check the grace period that is set, specify `mmrepquota -t`

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmsetquota` command.

GPFS must be running on the node from which the `mmsetquota` command is issued.

You may issue the `mmsetquota` command only from a node in the GPFS cluster where the file system is mounted.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see the topic *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

Examples

1. The following command sets the block soft and hard limit to 25G and 30G and files soft and hard limit to 10K and 11K, respectively for user user234:

```
# mmsetquota fs1 --user user234 --block 25G:30G --files 10K:11K
```

   To verify the change, issue the following command:

```
# mmlsquota -u user234 fs1
                     Block Limits                                    |     File Limits
Filesystem Fileset type      KB      quota    limit  in_doubt  grace |   files   quota limit in_doubt grace  Remarks
fs1        root     USR     143688   26214400 31457280   0     none |       1   10240  11264        0  none
fs1        mkfiles2 USR    no limits
fs1        ifset1   USR    no limits
fs1        1111     USR    no limits
fs1        ifset2   USR    no limits
```

2. If perfileset quota is enabled, the following command sets block soft and hard limit to 5G and 7G, respectively, for group fvt090 and for fileset ifset2:

```
# mmsetquota fs1:ifset2 --group fvt090 --block 5G:7G
```

To verify the change, issue the following command:

```
# mmlsquota -g fvt090 fs1

Disk quotas for group fvt090 (gid 2590):
                         Block Limits                                    |     File Limits
Filesystem Fileset  type   KB    quota   limit  in_doubt  grace |  files quota  limit in_doubt  grace  Remarks
fs1        root     GRP   none
fs1        mkfiles2 GRP   none
fs1        ifset1   GRP   none
fs1        1111     GRP   none
fs1        ifset2   GRP   143704 5242880 7340032   0     none |    1       0        0        0     none
```

3. To change the user grace period for block data to 10 days, issue the following command:

```
# mmsetquota fs1 --grace user --block 10days
```

4. All of the previous examples can be done in one invocation of `mmsetquota` by using quota stanza file. The stanza file `/tmp/quotaExample` may look like this:

```
%quota:
    device=fs1
    command=setquota
    type=USR
    id=user234
    blockQuota=25G
    blockLimit=30G
    filesQuota=10K
    filesLimit=11K

%quota:
    device=fs1
    command=setquota
    type=GRP
    id=fvt090
    fileset=ifset2
    blockQuota=5G
    blockLimit=7G

%quota:
    device=fs1
    command=setgraceperiod
    type=user
    blockGrace=10days
```

Then issue the command:

```
# mmsetquota –F /tmp/quotaExample
```

See also

- “mmcheckquota command” on page 220
- “mmdefedquota command” on page 345
- “mmdefquotaoff command” on page 349
- “mmdefquotaon command” on page 352
- “mmedquota command” on page 404
- “mmlsquota command” on page 535
- “mmquotaon command” on page 654
- “mmquotaoff command” on page 651
- “mmrepquota command” on page 666

Location

`/usr/lpp/mmfs/bin`

mmshutdown command

Unmounts all GPFS file systems and stops GPFS on one or more nodes.

Synopsis

```
mmshutdown [-t UnmountTimeout] [-a | -N {Node[,Node...] | NodeFile | NodeClass}] [--accept]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmshutdown` command to stop the GPFS daemons on one or more nodes. If no operand is specified, GPFS is stopped only on the node from which the command was issued.

The `mmshutdown` command first attempts to unmount all GPFS file systems. If the unmount does not complete within the specified *timeout* period, the GPFS daemons shut down anyway.

If shutting down the specified nodes will cause problems in the cluster, the command prompts for confirmation. These checks can be bypassed by setting the **`confirmShutdownIfHarmful`** parameter value. To bypass these checks, issue the following command:

```
mmchconfig confirmShutdonwIfHarmful=no
```

Currently, the command checks only whether the shutdown will cause loss of node quorum.

Results

Upon successful completion of the `mmshutdown` command, these tasks are completed:

- GPFS file systems are unmounted.
- GPFS daemons are stopped.

Parameters

-a
Stop GPFS on all nodes in a GPFS cluster.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Directs the `mmshutdown` command to process a set of nodes.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

This command does not support a *NodeClass* of `mount`.

--accept
Bypasses the user confirmation, if the command detects that shutting down the specified nodes will cause a harmful condition.

Options

-t *UnmountTimeout*
The maximum amount of time, in seconds, that the unmount command is given to complete. The default timeout period is equal to:

60 + 3 × number of nodes

If the unmount does not complete within the specified amount of time, the command times out and the GPFS daemons shut down.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the mmshutdown command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To stop GPFS on all nodes in the GPFS cluster, issue this command:

```
mmshutdown -a
```

The system displays information similar to:

```
Thu Mar 15 14:02:50 EDT 2012: mmshutdown: Starting force unmount of GPFS file systems
c6f1c3vp2.gpfs.net:  forced unmount of /gpfs/fs1
c6f1c3vp1.gpfs.net:  forced unmount of /gpfs/fs1
Thu Mar 15 14:03:00 EDT 2012: mmshutdown: Shutting down GPFS daemons
c6f1c3vp3.gpfs.net:  Shutting down!
c6f1c3vp4.gpfs.net:  Shutting down!
c6f1c3vp4.gpfs.net:  'shutdown' command about to kill process 23649
c6f1c3vp4.gpfs.net:  Unloading modules from /lib/modules/2.6.27.19-5-ppc64/extra
c6f1c3vp4.gpfs.net:  Unloading module mmfs26
c6f1c3vp1.gpfs.net:  Shutting down!
c6f1c3vp4.gpfs.net:  Unloading module mmfslinux
c6f1c3vp1.gpfs.net:  'shutdown' command about to kill process 7667944
c6f1c3vp2.gpfs.net:  Shutting down!
c6f1c3vp2.gpfs.net:  'shutdown' command about to kill process 5701764
c6f1c3vp2.gpfs.net:  Master did not clean up; attempting cleanup now
c6f1c3vp2.gpfs.net:  Thu Mar 15 14:04:05.114 2012: mmfsd is shutting down.
c6f1c3vp2.gpfs.net:  Thu Mar 15 14:04:05.115 2012: Reason for shutdown: mmfsadm shutdown command timed out
c6f1c3vp2.gpfs.net:  Thu Mar 15 14:04:06 EDT 2012: mmcommon mmfsdown invoked.  Subsystem: mmfs Status:
down
c6f1c3vp2.gpfs.net:  Thu Mar 15 14:04:06 EDT 2012: mmcommon: Unmounting file systems ...
Thu Mar 15 14:04:10 EDT 2012: mmshutdown: Finished
```

2. To stop GPFS on only node k164n04, issue this command:

```
mmshutdown -N k164n04
```

The system displays information similar to:

```
Thu Mar 15 14:00:12 EDT 2012: mmshutdown: Starting force unmount of GPFS file systems
k164n04:  forced unmount of /gpfs/fs1
Thu Mar 15 14:00:22 EDT 2012: mmshutdown: Shutting down GPFS daemons
k164n04:  Shutting down!
k164n04:  'shutdown' command about to kill process 7274548
Thu Mar 15 14:00:45 EDT 2012: mmshutdown: Finished
```

See also

- “mmgetstate command” on page 431
- “mmlscluster command” on page 492
- “mmstartup command” on page 726

Location

/usr/lpp/mmfs/bin

mmsmb command

Administers SMB shares, export ACLs, and global configuration.

Synopsis

```
mmsmb export list [ListofSMBExports ][ -Y ][ --option Arg ][ --export-regex Arg ]
[ --header N ][ --all ][ --key-info Arg]
```

or

```
mmsmb export add SMBExport Path [--option SMBOption=Value | --key-info SMBOption]
```

or

```
mmsmb export change SMBExport [--option SMBOption=Value | --remove SMBOption
|--key-info SMBOption]
```

or

```
mmsmb export remove SMBExport [--force]
```

or

```
mmsmb config list [ListofSMBOptions | -Y | --supported | --header N
| --key-info SMBOption]
```

or

```
mmsmb config change {--option SMBOption=Value | --remove SMBOption
|--key-info SMBOption | --vfs-fruit-enable}
```

or

```
mmsmb exportacl getid { Name | --user UserName | --group GroupName
| --system SystemName }
```

or

```
mmsmb exportacl list { [ExportName] | [ExportName --viewsddl] }[--viewsids]
```

or

```
mmsmb exportacl add ExportName { Name | --user UserName | --group GroupName
| --system SystemName | --SID SID } [--viewsids] --access Access --permissions
Permissions [--force]
```

or

```
mmsmb exportacl change ExportName { Name | --user UserName | --group GroupName
| --system SystemName | --SID SID } [--viewsids] --access Access --permissions Permissions
```

or

```
mmsmb exportacl remove ExportName { Name | --user UserName | --group GroupName
| --system SystemName | --SID SID } [--viewsids]  [--access Access ]
[--permissions Permissions ]
```

or

```
mmsmb exportacl replace ExportName { Name | --user UserName | --group GroupName
| --system SystemName | --SID SID } --access Access--permissions Permissions [--viewsids]
[--force]
```

or

```
mmsmb exportacl delete ExportName [--viewsids]  [--force]
```

Note: For **mmsmb export**, you can specify ––option ––remove multiple times but you cannot specify both of these options simultaneously.

Availability

Available on all IBM Spectrum Scale editions.

The protocol functions provided in this command, or any similar command, are generally referred to as CES (Cluster Export Services). For example, protocol node and CES node are functionally equivalent terms.

Description

Use the mmsmb command to administer SMB shares and global configuration.

Using the mmsmb export command, you can do the following tasks:

- Create the specified SMB share. The mmsmb export add command creates the specified export for the specified path. Any supported SMB option can be specified by repeating --option. Also the substitution values %D for the domain, %U for session user name and %G for the primary group of %U are supported as part of the specified path. The % character is not allowed in any other context. If the export exists but the path does not exist or if it is not inside the GPFS file system, the command returns with an error. When one or more substitution variables are used, only the sub-path to the first substitution variable is checked. If authentication on the cluster is not enabled, this command will terminate with an error.
- Change the specified SMB share using the mmsmb export change command.
- Delete an SMB share using the mmsmb export remove command. Existing connections to the deleted SMB share will be disconnected. This can result in data loss for files being currently open on the affected connections.
- List the SMB shares by using the mmsmb export list command. The command displays the configuration of options for each SMB share. If no specific options are specified, the command displays all SMB shares with the SMB options browseable; guest ok; smb encrypt as a table. Each row represents an SMB share and each column represents an SMB option.

Using the mmsmb config command, you can do the following tasks:

- Change, add or remove the specified SMB option for the SMB configuration. Use the mmsmb config change command to change the global configuration.
- List the global configuration of SMB shares. Use the mmsmb config list command to display the global configuration options of SMB shares. If no specific options are specified, the command displays all SMB option-value pairs.
- Enable SMB fruit support

Using the mmsmb exportacl command, you can do the following tasks:

- Retrieve the ID of the specified user/group/system.
- List, change, add, remove, replace and delete the ACL associated with an export.
- The add option has two mandatory arguments: --access and --permissions.

Parameters

mmsmb export

list
Lists the SMB shares.

ListofSMBExports
Specifies the list of SMB shares that needs to be listed as blank separated strings.

--option {*key* | all | unsupported}

key
Specifies only the supported SMB option to be listed.

all
Displays all used SMB options.

unsupported
Detects and displays all SMB shares with unsupported SMB options. The path of all unsupported options are listed for each SMB share.

--export-regex *arg*
arg is a regular expression against which the exportnames are matched. Only matching exports are shown. If this option is not specified and if *ListofSMBExports* are also not specified, all existing exports are displayed.

--all
Displays all defined SMB options. Similar to `--option all`.

add
Creates the specified SMB share on a GPFS file system with NFSv4 ACLs enforced. You can verify whether your GPFS files system has been configured correctly by using the **mmlsfs** command. For example, `mmlsfs gpfs0 -k`.

path
Specifies the path of the SMB share that needs to be added.

--option *SMBoption=value*
Specifies the SMB option for the SMB protocol. If it is not a supported SMB option or the value is not allowable for this SMB option, the command terminates with an error. If this option is not specified, the default options: `guest ok = no` and `smb encrypt = auto` are set.

Note: You cannot change the "guest ok" option by using the **mmsmb** command as it is an unsupported option.

change
Modifies the specified SMB share.

--option *SMBOption=value*
Specifies the SMB option for the SMB protocol. If the SMB option is not configured for the specified export, it will be added with the specified value. If the SMB option is not supported or the value is not allowable for this SMB option the command terminates with an error. If no value is specified, the specified SMB option is set to default by removing the current setting from the configuration.

`--remove` *SMBOption*
Specifies the SMB option that is to be removed. If the SMB option is supported it will be removed from the specified export. The default value becomes active. If the SMB option is not supported, the command terminates with an error.

`--vfs-fruit-enable`
Enables the `vfs_fruit` module and alternate data streams support for better support of Mac OS SMB2 clients. Setting this option requires the SMB service to be down on all CES nodes. Disabling it requires the help of IBM support as the Apple file meta-data that has been moved to extended attributes and would not be accessible any more unless restored back to files. For more information, see the Support of *vfs_fruit* topic of the *IBM Spectrum Scale: Administration Guide*.

remove
Deletes the specified SMB share.

--force
Suppresses confirmation questions.

SMBExport
Specifies the SMB share that needs to be listed.

List of supported SMB options for the `mmsmb export {list | add | change | remove}` command:

admin users
Using this option, administrative users can be defined in the format of `admin users=user1,user2,..,usern`. This is a list of users who will be granted administrative privileges on the share. This means that they will do all file operations as the super user (root). You should use this option very carefully, as any user in this list will be able to do anything they like on the share, irrespective of file permissions.

Default: `admin users =`

Example: `admin users = win-dom\jason`

browseable
If the value is set as `yes`, the export is shown in the Windows Explorer browser when browsing the file server. By default, this option is enabled.

comment
Description of the export.

csc policy
`csc policy` stands for client–side caching policy, and specifies how clients that are capable of offline caching cache the files in the share. The valid values are: `manual` and `disable`. Setting `csc policy = disable` disables offline caching. For example, this can be used for shares containing roaming profiles. By default, this option is set to the value `manual`.

fileid:algorithm
This option allows to control the level of enforced data integrity. If the data integrity is ensured on the application level, it can be beneficial in cluster environments to reduce the level of enforced integrity for performance reasons.

`fsname` is the default value that ensures data integrity in the entire cluster by managing concurrent access to files and directories cluster-wide.

The `fsname_norootdir` value disables synchronization of directory locks for the root directory of the specified export only and keeps locking enabled for all files and directories within and underneath the share root.

The `fsname_nodirs` value disables synchronization of directory locks across the cluster nodes, but keeps locking enabled for files.

The `hostname` value completely disables cross–node locking for both directories and files on the selected share.

Note: Data integrity is ensured if an application does not use multiple processes to access the data at the same time, for example, reading of file content does not happen while another process is still writing to the file. Without locking, the consistency of files is no longer guaranteed on protocol level. If data integrity is not ensured on application level this can lead to data corruption. For example, if two processes modify the same file in parallel, assuming that they have exclusive access.

gpfs:leases
`gpfs:leases` are cross protocol `oplocks` (opportunistic locks), that means an SMB client can lock a file that provides the user improved performance while reading or writing to the file because no other user read or write to this file. If the value is set as `yes`, clients accessing the file over the other protocols can break the oplock of an SMB client and the user gets informed when another user is accessing the same file at the same time.

gpfs:recalls
If the value is set as `yes` files that have been migrated from disk will be recalled on access. By default, this is enabled. If `recalls = no` files will not be recalled on access and the client will receive `ACCESS_DENIED` message.

gpfs:sharemodes
An application can set share modes. If you set `gpfs:sharemodes = yes`, using the `mmsmb export change` *SMBexport* `--option "gpfs:sharemodes = yes"` the **sharemodes** specified by the application will be respected by all protocols and not only by the SMB protocol. If you set `gpfs:sharemodes = no` the **sharemodes** specified by the application will only be respected by the SMB protocol. For example, the NFS protocol will ignore the **sharemode** set by the application.

The application can set the following **sharemodes**: `SHARE_READ` or `SHARE_WRITE` or `SHARE_READ and SHARE_WRITE` or `no sharemodes`.

gpfs:syncio
If the value is set as `yes`, it specifies the files in an export, for which the setting is enabled, are opened with the O_SYNC flag. Accessing a file is faster if **gpfs:syncio** is set to `yes`.

Performance for certain workloads can be improved when SMB accesses the file with the `O_SYNC` flag set. For example, updating only small blocks in a large file as observed with database applications. The underlying GPFS behavior is then changed to not read a complete block if there is only a small update to it. By default, this option is disabled.

hide unreadable
If the value is set as `yes`, all files and directories that the user has no permission to read are hidden from directory listings in the export. The `hideunreadable=yes` option is also known as access–based enumeration because when a user is listing (enumerating) the directories and files within the export, they only see the files and directories that they have read access to. By default, this option is disabled.

Warning: Enabling this option has a negative impact on directory listing performance, especially for large directories as the ACLs for all directory entries have to be read and evaluated. This option is disabled by default.

oplocks
If the value is set as `yes`, a client may request an opportunistic lock (**oplock**) from an SMB server when it opens a file. If the server grants the request, the client can cache large chunks of the file without informing the server what it is doing with the cached chunks until the task is completed. Caching large chunks of a file saves a lot of network I/O round–trip time and enhances performance. By default, this option is enabled.

Warning: While **oplocks** can enhance performance, they can also contribute to data loss in case of SMB connection breaks/timeouts. To avoid the loss of data in case of an interface node failure or storage timeout, you might want to disable **oplocks**.

Opportunistic locking allows a client to notify the SMB server that it will be the exclusive writer of the file. It also notifies the SMB server that it will cache its changes to that file on its own system and not on the SMB server to speed up file access for that client. When the SMB server is notified about a file being opportunistically locked by a client, it marks its version of the file as having an opportunistic lock and waits for the client to complete work on the file. The client has to send the final changes back to the SMB server for synchronization. If a second client requests access to that file before the first client has finished working on it, the SMB server can send an `oplock` break request to the first client. This request informs the client to stop caching its changes and return the current state of the file to the server so that the interrupting client can use it. An opportunistic lock, however, is not a replacement for a standard `deny-mode` lock. There are many use cases when the interrupting process to be granted an `oplock` break only to discover that the original process also has a `deny-mode` lock on the file.

posix locking
If the value is set as `yes`, it will be tested if a byte–range (`fcntl`) lock is already present on the requested portion of the file before granting a byte–range lock to an SMB client. For improved performance on SMB–only shares this option can be disabled, which is the default behavior. Disabling `locking` on cross–protocol shares can result in data integrity issues when clients concurrently set locks on a file via multiple protocols, for example, SMB and NFS.

read only
If the value is set as `yes`, files cannot be modified or created on this export independent of the ACLs. By default, the value is `no`.

smb encrypt
This option controls whether the remote client is allowed or required to use SMB encryption. Possible values are `auto`, `mandatory`, `disabled`, and `desired`. This is set when the export is created with default value, which is `auto`. Clients may chose to encrypt the entire session, not just traffic to a specific export. The server would return `access denied` message to all non–encrypted requests on such an export. Selecting encrypted traffic reduces throughput as smaller packet sizes must be used as well as the overhead of encrypting and signing all the data. If SMB encryption is selected, the message integrity is guaranteed so that signing is implicit. When set to `auto`, SMB encryption is offered, but not enforced. When set to `mandatory`, SMB encryption is required and if set to disabled, SMB encryption cannot be negotiated.

This setting controls SMB encryption setting for the individual SMB share. Supported values are:

- `auto/default`: This will enable negotiation of encryption but will not turn on data encryption globally.
- `mandatory`: This will enable SMB encryption and turn on data encryption for this share. Clients that do not support encryption will be denied access to this SMB share.

  Note: This requires the global `mmsmb config smb encrypt` setting to be either `auto/default` or `mandatory`.
- `disabled`: Disables the encryption feature for a specific SMB share.
- `desired`: Enables negotiation and turns on data encryption on sessions and share connections for those clients that support it.

syncops:onclose
This option ensures that the file system synchronizes data to the disk each time a file is closed after writing. The data written is flushed to the disk, but this can reduce performance for workloads writing many files. Enabling this option will decrease the risk of possible data loss in case of a node failure. This option is disabled by default. Enabling this should not be necessary, as data is flushed to disk periodically by the file system and also when requested from an application on a SMB client.

Default: "syncops:onclose = no"

Allowable values: yes, no

Example: "syncops:onclose = yes"

Warning: Enabling this option can have a negative performance impact on workloads creating many files from an SMB client.

hide dot files
Setting this to "no" will remove hidden attributes for dot files in an SMB share so that those files become visible when you access that share.

msdfs proxy
This option enables the IBM Spectrum Scale customers to configure DFS redirects for SMB shares. If an SMB client attempts to connect to such a share, the client is redirected to one or multiple proxy shares by using the SMB-DFS protocol.

Here, "msdfs proxy" needs a value to be assigned, which is the list of proxy shares. The syntax for this list is,

```
\<smbserver1>\<someshare>[,\<smbserver2>\<someshare>,...]
```

mmsmb config

list
Lists the global configuration options of SMB shares.

ListofSMBOptions
Specifies the list of SMB options that needs to be listed.

--supported
Displays all changeable SMB options and their values.

change
Modifies the global configuration options of SMB shares.

--option *SMBOption=value*
Sets the value of the specified SMB option. If no value is given, the SMB option is removed.

Specifies the SMB option for the SMB protocol. If the SMB option is not configured for global configuration, it will be added with the specified value. If no value is specified, the specified SMB option is set to default and removed from the global configuration. If the SMB option is not supported or the value is not allowable for the SMB option, the command terminates with an error.

--remove *SMBOption*
Specifies the SMB option that is to be removed. If the SMB option is supported it will be removed from the global configuration. The default value becomes active. If the SMB option is not supported, the command terminates with an error.

List of supported SMB options by the `mmsmb config {list | change}` command:

gpfs:dfreequota
`gpfs:dfreequota` stands for disk Free Quota. If the value is set to `yes` the free space and size reported to a SMB client for a share will be adjusted according to the applicable quotas. The applicable quotas are the quota of the user requesting this information, the quota of the user's primary group and the quota of the fileset containing the export.

restrict anonymous
The setting of this parameter determines whether access to information is allowed or restricted for anonymous users. The options are:

`restrict anonymous = 2`: anonymous users are restricted from accessing information. This is the default setting.

`restrict anonymous = 0`: anonymous users are allowed to access information.

`restrict anonymous = 1`: is not supported

server string
`server string` stands for Server Description. It specifies the server description for SMB protocol. Server description with special characters must be provided in single quotes.

smb encrypt

This setting controls the global SMB encryption setting that applies to each SMB connection. Supported values are:

- `auto/default`: This will enable negotiation of encryption but will not turn on data encryption globally.
- `mandatory`: This will enable negotiation and turn on data encryption for all SMB shares. Clients that do not support encryption will be denied access to the server.
- `disabled`: This will completely disable the encryption feature for all connections.

  Note: With `smb encrypt = disabled` on the share and `smb encrypt = mandatory` in the global section, access will be denied for all clients.

mmsmb exportacl

getid
Retrieve the ID of the specified user/group/system.

```
mmsmb exportacl getid myUser
mmsmb exportacl getid --user myUser
```

```
mmsmb exportacl getid --group myGroup
mmsmb exportacl getid --system mySystem
```

list

List can only take viewing options.

```
mmsmb exportacl list myExport               Show the export ACL for this exportname
mmsmb exportacl list                        Show all the export ACLs
mmsmb exportacl list myExport --viewsddl    Show the export ACL for this exportname in
sddl format.
```

add

Add will add a new permission to the export ACL. It will include adding a user, group or system. The options are:

- {User/group/system name} If you do not specify the type of name, the system will prioritize in this order:
- --user
- --group
- --system, or
- --SID (this can be the SID for a user, group or system).

Mandatory arguments are:

- --access: ALLOWED or DENIED
- --permissions: One of FULL, CHANGE, or READ or any combination of RWXDPO.

Examples:

```
mmsmb exportacl add myExport0 myUser --access ALLOWED --permissions FULL
mmsmb exportacl add myExport1 --user user01 --access ALLOWED --permissions RWO
mmsmb exportacl add myExport2 --group   group01 --access DENIED -- permissions
RWXDP
```

change

Change will update the specified ACE in an export ACL. The options are:

- {User/group/system name} If you do not specify the type of name, the system will prioritize in this order:
- --user
- --group
- --system
- --SID (This can be the SID for a user, group or system).

Mandatory arguments are:

- --access: ALLOWED or DENIED
- --permissions: One of FULL, CHANGE, or READ or any combination of RWXDPO.

Examples:

```
mmsmb exportacl change myExport --user myUser --access ALLOWED --permissions RWX
mmsmb exportacl change myExport --group allUsers --access ALLOWED --permissions R
```

remove

Remove will remove the ACE for the specified user/group/system from the ACL.

The user, group or system will be removed automatically for a specified name.

In the event that the system is unable to locate an ACE within the export ACL that it can remove the permissions for as instructed, an error will be issued to the user informing them of this.

The options are:

- {User/group/system name} If you do not specify the type of name, the system will prioritize in this order:
- --user
- --group
- --system
- --SID (This can be the SID for a user, group or system).

Optional arguments are:

- --access: ALLOWED or DENIED
- --permissions: One of FULL, CHANGE, or READ or any combination of RWXDPO.

Examples:

```
mmsmb exportacl remove myExport01 UserName --access ALLOWED --permissions
CHANGE
mmsmb exportacl remove myExport02 GroupName --access DENIED --permissions READ
mmsmb exportacl remove myExport03 UserName
```

replace

The replace command replaces all the permissions in a export ACL with those indicated in its ACE specification. It is therefore a potentially destructive command and will include a confirmation. This confirmation can be overridden with the --force command. The options are:

- {User/group/system name} If you do not specify the type of name, the system will prioritize in this order:
- --user
- --group
- --system
- --SID (This can be the SID for a user, group or system)
- --force.

Mandatory arguments are:

- --access: ALLOWED or DENIED
- --permissions: One of FULL, CHANGE, or READ or any combination of RWXDPO.

Examples:

```
mmsmb exportacl replace myExport01 --user user01 --access ALLOWED --permissions FULL
mmsmb exportacl replace myExport02 --group group01 --access ALLOWED --permissions READ
--force
mmsmb exportacl replace myUser --access ALLOWED --permissions FULL
```

delete

The Delete command will remove an entire export ACL. It therefore does not require a system, id or user identified as they are not appropriate in this case. All it needs is the name of an export for which the export ACL will be deleted.

Delete will include a confirmation. This can be overridden using the --force parameter.

Examples:

```
mmsmb exportacl delete myExport01
mmsmb exportacl delete myExport02 --force
```

Parameters common for both `mmsmb export` and `mmsmb config` commands

-Y

Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

--header *n*
Repeats the output table header every *n* lines for a table that is spread over multiple pages. The value *n* can be of any integer value.

--key-info *arg*
Displays the supported SMB options and their possible values.

arg SMBoption | supported

SMBoption
Specifies the SMB option.

supported
Displays descriptions for all the supported SMB options.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the mmsmb command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

mmsmb config list

1. Show descriptions of all the supported SMB configuration options.

```
mmsmb config list --key-info supported
```

The system displays output similar to this:

```
Supported smb options with allowed values:
gpfs:dfreequota    = yes, no
restrict anonymous = 0, 2
server string      = any
```

2. List the SMB option that specifies whether anonymous access is allowed or not.

```
mmsmb config list "gpfs:dfreequota"
```

The system displays output similar to this:

```
SMB option        value
gpfs:dfreequota   yes
```

3. Display the SMB configuration options in a machine-readable format.

```
mmsmb config list -Y
```

The system displays output similar to this:

```
add share command:aio read size:aio write size:aio_pthread%3Aaio open:async smb echo handler:auth
methods:change
notify:change share command:client NTLMv2 auth:ctdb locktime warn threshold:debug hires timestamp:delete
share
command:dfree cache time:disable netbios:disable spoolss:dmapi support:ea support:fileid%3Amapping:force
unknown
acl
user:gencache%3Astabilize_count:gpfs%3Adfreequota:gpfs%3Ahsm:gpfs%3Aleases:gpfs%3Aprealloc:gpfs%3Asharemo
des:
```

mmsmb config change

1. Show descriptions of all the supported SMB configuration options.

   ```
   mmsmb config change --key-info supported
   ```

   The system displays output similar to this:

   ```
   Supported smb options with allowed values:
   gpfs:dfreequota    = yes, no
   restrict anonymous = 0, 2
   server string      = any
   ```

   Note: The output of this command depends on the configuration options supported by the system.

2. Change an SMB configuration option.

   ```
   mmsmb config change --option "restrict anonymous=0"
   ```

   You can confirm the change by using this `mmsmb config list "restrict anonymous"` command.

3. Remove an SMB configuration option.

   ```
   mmsmb config change --remove "server string"
   ```

   The system displays output similar to this:

   ```
   Warning:
           Unused options suppressed in display:
                   server string
   ```

mmsmb export list

1. To list all SMB options for export `myExport`, issue this command:

   ```
   mmsmb export list myExport --option all
   ```

   The system displays output similar to this:

   ```
   export          path              smb          encrypt       guest ok
   browseable
   myExport        /gpfs/fs0/combo-1   auto      no            no
   ```

2. To list SMB option `csc policy` for SMB share `myexport` and all SMB shares starting with `foo` followed by any quantity of 1 and ending on 2 or 3, issue this command:

   ```
   mmsmb export list --option "csc policy" myexport --export-regex "foo1*[23]$"
   ```

   The system displays output similar to this:

   ```
   export      path                                csc policy
   myExport    /mnt/gpfs0/shared                    - - -
   foo11b23    /mnt/gpfs0/testExport/testSmbEncypt   disable
   foo111b23   /mnt/gpfs0/testExport/testSmbEncypt   documents
   ```

3. To list all exports where unsupported SMB options are set, issue this command:

   ```
   mmsmb export list --option unsupported
   ```

The system displays output similar to this:

```
export                   path                    mangled names
hasForbiddenOptions     /mnt/gpfs0/shared       yes
```

4. To list the DFS redirect option setting for a share, issue this command:

```
mmsmb export list redirect --option "msdfs proxy"
```

The system displays output similar to this:

```
export     path                    msdfs proxy
redirect   /ibm/gpfs0/redirect/    \cluster1\share
```

5. To list all share options (including "msdfds proxy" setting), issue this command:

```
mmsmb export list redirect --all
```

The system displays output similar to this:

```
export   path                smb encrypt msdfs proxy     guest ok comment                   browseable
redirect /ibm/gpfs0/redirect/ auto        \cluster1\share no       redirect to /cluster1/share no
```

mmsmb export add

1. To create an export `myExport` as default SMB share for path `/ibm/gpfs0/myFolder` with default options, issue this command:

```
mmsmb export add myExport "/ibm/gpfs0/myFolder"
```

The system displays output similar to this:

```
mmsmb export add: The SMB export was created successfully.
```

2. To add a new share and set "msdfs proxy" share option, issue this command:

```
msmb export add redirect /ibm/gpfs0/redirect --option "msdfs proxy"="/cluster1/share"
```

The system displays output similar to this:

```
mmsmb export add: The SMB export was created successfully.
```

mmsmb export change

1. To change the SMB option `oplocks` to value `yes` for SMB share `myExport`, issue this command:

```
mmsmb export change myExport --option "oplocks"="yes"
```

You can confirm the change by using this **mmsmb export list --option "oplocks" myExport** command. The system displays output similar to this:

```
export     path                oplocks
myExport   /mnt/gpfs0/shared   yes
```

2. To remove the SMB option `oplocks` from SMB share `myExport`, issue the following commands:

```
mmsmb export change myExport --remove "oplocks"
```

You can confirm the change by using this **mmsmb export list --option all myExport** command.

3. To change the "msdfs proxy" share option, issue this command:

```
mmsmb export change redirect --option "msdfs proxy"="\cluster1\share"
```

The system displays output similar to this:

```
mmsmb export change: The SMB export was changed successfully.
```

mmsmb export remove

1. To delete the SMB share myExport with confirmation, issue this command:

   ```
   mmsmb export remove myExport
   ```

   The system asks for confirmation, similar to this:

   ```
   Do you really want to perform the operation (yes/no - default no):
   ```

2. To delete the SMB share myExport without confirmation:

   ```
   mmsmb export remove myExport --force
   ```

   The system removes the SMB share without any confirmation.

See also

- “mmnfs command” on page 560
- “mmces command” on page 133
- “mmlsfs command” on page 506

Location

/usr/lpp/mmfs/bin

mmsnapdir command

Controls how the special directories that connect to snapshots appear.

Synopsis

```
mmsnapdir Device [-r | -a]
    [--show-global-snapshots {rootfileset | allfilesets}]
    [{[--fileset-snapdir FilesetSnapDirName] [--global-snapdir GlobalSnapDirName]}
     | {{--snapdir | -s} SnapDirName}]
```

or

```
mmsnapdir Device [-q]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmsnapdir` command to control how the special directories that connect to snapshots appear. Both the name of the directories and where they are accessible can be changed.

Global snapshots appear in a subdirectory in the root directory of the file system, whose default name is `.snapshots`. Fileset snapshots appear in a similar `.snapshots` subdirectory located in the root directory of each independent fileset. These special subdirectories are collectively referred to as *snapdirs*. Note that the root directory of the file system and the root directory of the root fileset are the same, so global snapshots and fileset snapshots of the root fileset will both appear in the same snapdir.

If you prefer to access the snapshots from each directory rather than traversing through the root directory, you can use an invisible directory to make the connection by issuing the `mmsnapdir` command with the `-a` option (see "Examples"). The `-a` option enables an invisible directory in each directory in the active file system (they do not appear in directories in snapshots) that contains a subdirectory for each existing snapshot of the file system (in the root fileset) or fileset (in other independent filesets). These subdirectories correspond to the copy of the active directory in the snapshot with the same name. For example, if you enter `ls -a /fs1/userA`, the (invisible) `.snapshots` directory is not listed. However, you can use `ls /fs1/userA/.snapshots`, for example, to confirm that `.snapshots` is present and contains the snapshots holding copies of `userA`. When the `-a` option is enabled, the paths `/fs1/.snapshots/Snap17/userA` and `/fs1/userA/.snapshots/Snap17` refer to the same directory, namely `userA` at the time when `Snap17` was created. The `-r` option (root-directories-only), which is the default, reverses the effect of the `-a` option (all-directories), and disables access to snapshots via snapdirs in non-root directories.

If you prefer to access global snapshots from the root directory of all independent filesets, use the `mmsnapdir` command with the `--show-global-snapshots allfilesets` option. With this option, global snapshots will also appear in the snapdir in the fileset root directory. The global snapshots will also appear in the snapdirs in each non-root directory if all-directories (the `-a` option) is enabled. To return to the default setting use `--show-global-snapshots rootfileset`, and global snapshots will only be available in root of the file system, or the root fileset, if all-directories is enabled.

The name of the snapdir directories can be changed using the `--snapdir` (or `-s`) option. This name is used for both global and fileset snapshots in both fileset root directories and, if all-directories is enabled, non-root directories also. The snapdir name for global and fileset snapshots can be specified separately using the `--global-snapdir` and `--fileset-snapdir` options. If these names are different, two snapdirs will appear in the file system root directory, with the global and fileset snapshots listed separately. When `--show-global-snapshots` is set to `allfilesets`, two snapdirs will appear in fileset root directories also, and when all-directories (the `-a` option) is specified, the two snapdirs will be available in non-root directories as well. If `--global-snapdir` is specified by itself, the fileset snapdir

name is left unchanged, and vice versa if `--fileset-snapdir` option is used. Setting both snapdirs to the same name is equivalent to using the `--snapdir` option. The snapdir name enabled in non-root directories by all-directories is always the same as the name used in root directories.

For more information on global snapshots, see *Creating and maintaining snapshots of file systems* in the *IBM Spectrum Scale: Administration Guide*.

For more information on fileset snapshots, see *Fileset-level snapshots* in the *IBM Spectrum Scale: Administration Guide*.

Parameters

Device
The device name of the file system. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

This must be the first parameter.

-a
Adds a snapshots subdirectory to all subdirectories in the file system.

-r
Reverses the effect of the `-a` option. All invisible snapshot directories are no longer accessible. The snapshot directory under the file system root directory is not affected.

--show-global-snapshots {rootfileset | allfilesets}
This option controls whether global snapshots are accessible through a subdirectory under the root directory of all independent filesets (`allfilesets`) or only in the file system root (`rootfileset`). For example, issuing the following command:

```
mmsnapdir fs1 --show-global-snapshots allfilesets
```

specifies that the root directory of each independent fileset will contain a `.gsnaps` subdirectory listing all global snapshots, such as `/fs1/junctions/FsetA/.gsnaps` and `/fs1/junctions/FsetA/.fsnaps`. This can be used to make global snapshots accessible to clients, for example NFS users, that do not have access to the file system root directory.

Specifying `rootfileset` reverses this feature, restoring the default condition, so that global snapshots are only visible in the root fileset.

--fileset-snapdir *FilesetSnapDirName*
--global-snapdir *GlobalSnapDirName*
The `--global-snapdir` option specifies the name for the directory where global snapshots are listed. The `--fileset-snapdir` option specifies the name for the directory where fileset snapshots are listed. These options can be specified together or separately, in which case only the corresponding snapdir is changed. Neither option may be specified with `--snapdir`, which sets both to the same name.

For example, after issuing the command:

```
mmsnapdir fs1 --fileset-snapdir .fsnaps --global-snapdir .gsnaps
```

the directory `/fs1/.gsnaps` will list all global snapshots and `/fs1/.fsnaps` will only list fileset snapshots of the root fileset. Fileset snapshots of other independent filesets will be listed in `.fsnaps` under the root directory of each independent fileset, such as `/fs1/junctions/FsetA/.fsnaps`.

--snapdir | -s *SnapDirName*
Changes the name of the directory for both global and fileset snapshots to *SnapDirName*. This affects both the directory in the file system root as well as the invisible directory in the other file system directories if the `-a` option has been enabled. The root and non-root snapdirs cannot be given different names.

-q
Displays current snapshot settings. The `-q` option cannot be specified with any other options. This is the default if no other options are specified.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

If you are a root user, the node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see the topic *Requirements for administering a GPFS file system* in the *IBM Spectrum Scale: Administration Guide*.

You must be a root user to use all of the `mmsnapdir` options. Non-root users can only use the `-q` option.

If you are a non-root user, you may only specify file systems that belong to the same cluster as the node on which the `mmsnapdir` command was issued.

Examples

1. To rename the `.snapshots` directory (the default snapshots directory name) to `.link` for file system `fs1`, issue the command:

   ```
   mmsnapdir  fs1 -s .link
   ```

   After the command has been issued, the directory structure would appear similar to:

   ```
   /fs1/file1
   /fs1/userA/file2
   /fs1/userA/file3

   /fs1/.link/snap1/file1
   /fs1/.link/snap1/userA/file2
   /fs1/.link/snap1/userA/file3
   ```

2. To add the `.link` subdirectory to all subdirectories in the file system, issue:

   ```
   mmsnapdir fs1 -a
   ```

   After the command has been issued, the directory structure would appear similar to:

   ```
   /fs1/file1
   /fs1/userA/file2
   /fs1/userA/file3
   /fs1/userA/.link/snap1/file2
   /fs1/userA/.link/snap1/file3

   /fs1/.link/snap1/file1
   /fs1/.link/snap1/userA/file2
   /fs1/.link/snap1/userA/file3
   ```

   The `.link` subdirectory under the root directory and under each subdirectory of the tree provides two different paths to each snapshot copy of a file. For example, `/fs1/userA/.link/snap1/file2` and `/fs1/.link/snap1/userA/file2` are two different paths that access the same snapshot copy of `/fs1/userA/file2`.

3. To reverse the effect of the previous command, issue:

   ```
   mmsnapdir fs1 -r
   ```

   After the command has been issued, the directory structure would appear similar to:

   ```
   /fs1/file1
   /fs1/userA/file2
   /fs1/userA/file3

   /fs1/.link/snap1/file1
   ```

```
/fs1/.link/snap1/userA/file2
/fs1/.link/snap1/userA/file3
```

4. To display the current snapshot settings, issue:

```
mmsnapdir fs1 -q
```

The system displays output similar to:

```
Snapshot directory for "fs1" is ".link" (root directory only)
```

If there are independent filesets, fileset snapshots, or the global and fileset snapshot directory names are different in the file system, the system displays output similar to:

```
Fileset snapshot directory for "fs1" is ".link" (root directory only)
Global snapshot directory for "fs1" is ".link" in root directory only
```

See also

- “mmcrsnapshot command” on page 340
- “mmdelsnapshot command” on page 383
- “mmlssnapshot command” on page 540
- “mmrestorefs command” on page 675

Location

/usr/lpp/mmfs/bin

mmstartup command

Starts the GPFS subsystem on one or more nodes.

Synopsis

```
mmstartup [-a | -N {Node[,Node...] | NodeFile | NodeClass}] [-E EnvVar=value ...]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmstartup` command to start the GPFS daemons on one or more nodes. If no operand is specified, GPFS is started only on the node from which the command was issued.

Parameters

-a
Start GPFS on all nodes in a GPFS cluster.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Directs the `mmstartup` command to process a set of nodes.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

This command does not support a *NodeClass* of `mount`.

-E *EnvVar=value*
Specifies the name and value of an environment variable to be passed to the GPFS daemon. You can specify multiple `-E` options.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmstartup` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

To start GPFS on all nodes in the GPFS cluster, issue this command:

```
mmstartup -a
```

The system displays information similar to:

```
Thu Aug 12 13:22:40 EDT 2004: 6027-1642 mmstartup: Starting GPFS ...
```

See also

- “mmgetstate command” on page 431
- “mmlscluster command” on page 492
- “mmshutdown command” on page 706

Location

`/usr/lpp/mmfs/bin`

mmtracectl command

Sets up and enables GPFS tracing.

Synopsis

```
mmtracectl { --start | --stop | --off | --set | --status}
           [--trace={io | all | def | "Class Level [Class Level ...]" }]
           [--trace-recycle={off | local | global | globalOnShutdown }]
           [--aix-trace-buffer-size=BufferSize]
           [--tracedev-buffer-size=BufferSize]
           [--trace-file-size=FileSize] [--trace-dispatch={yes | no }]
           [--tracedev-compression-level=Level]
           [--tracedev-write-mode={blocking | overwrite }]
           [--tracedev-timeformat={relative | absolute | calendar }]
           [--tracedev-overwrite-buffer-size=Size]
           [--format | --noformat]
           [-N {Node [,Node...] | NodeFile | NodeClass }]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Attention: Use this command only under the direction of the IBM Support Center.

Use the `mmtracectl` command to perform the following functions:

- Start or stop tracing.
- Display the tracing status for the specified nodes.
- Turn tracing on (start or set trace recycle) or off on the next session. This is a persistent setting to automatically start trace each time GPFS starts.
- Allow for predefined trace levels: **io**, **all**, and **def**, as well as user-specified trace levels.
- Allow for changing the size of trace buffer sizes for AIX and all others using the **tracedev** option.
- Trace recycle functions, which allow for never cycling traces (`off` option), cycling traces on all nodes when GPFS ends abnormally (`global` option), and cycling traces any time GPFS goes down on all nodes (`globalOnShutdown` option).
- For Linux nodes only, this command allows you to change:
  - The trace writing mode
  - The raw data compression level

Note: Tracing on Windows requires support programs provided by Microsoft. For details about this prerequisite, see the section about configuring Windows and installing tracing support programs in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Results

GPFS tracing can be started, stopped, or related configuration options can be set.

Parameters

--start | --stop | --off | --set | --status
Specifies the actions that the `mmtracectl` command performs, where:

--start
Starts the trace.

--stop
Stops the trace.

--off
Clears all of the setting variables and stops the trace.

--set
Sets the trace variables.

--status
Displays the tracing status of the specified nodes. This parameter is rejected with a usage error message unless all the nodes in the cluster are running IBM Spectrum Scale 5.0.0 or later.

In the following table, the letter *X* indicates that the specified type of information is displayed:

Table 32. Tracing status information

Environment	Type of information that is displayed		
	Whether tracing is on or off	Kernel mode	Daemon mode
Linux	X	X	X
Windows	X		
AIX	X		

--trace={io | all | def | "*Class Level* [*Class Level* ...]"}
Allows for predefined and user-specified trace levels, where:

io
Indicates trace-level settings tailored for input and output (I/O).

all
Sets trace levels to their highest setting (9).

def
Indicates that the default trace settings will be used.

"*Class Level* [*Class Level* ...]"
Specifies a trace class and level.

--trace-recycle={off | local | global | globalOnShutdown}
Controls trace recycling during daemon termination. The following values are recognized:

off
Does not recycle traces. This is the default value until `mmtracectl --start` is run. If no trace-recycle value has been explicitly set when `mmtracectl --start` is run, see the following description for `local`. The `mmtracectl --off` command will remove any explicit value for `trace-recycle`, thus effectively setting `trace-recycle` back to the `off` value.

local
Recycles traces on the local node when **mmfsd** goes down abnormally. This setting also starts traces automatically any time that `mmstartup` is run to start the GPFS daemon, which could include an autoload on a reboot. If there is no `trace-recycle` value explicitly set at the time that `mmtracectl --start` is run, `mmchconfig` will be run to explicitly set the `trace-recycle` value to `local`. The starting of the actual tracing will be delayed while that configuration change is being made.

global
Recycles traces on all nodes in the cluster when an abnormal daemon shutdown occurs.

globalOnShutdown
Recycles traces on all nodes in the cluster for normal and abnormal daemon shutdowns.

--aix-trace-buffer-size=*BufferSize*
Controls the size of the trace buffer in memory for AIX.

--tracedev-buffer-size=*BufferSize*
Specifies the trace buffer size for Linux trace in blocking mode. If `--tracedev-write-mode` is set to blocking, this parameter will be used. It should be no less than 4K and no more than 64M. The default is 4M.

Note: This option applies only to Linux nodes.

--trace-file-size=*FileSize*
Controls the size of the trace file. The default is 128M on Linux and 64M on other platforms.

--trace-dispatch={yes | no}
Enables AIX thread dispatching trace hooks.

--tracedev-compression-level=*Level*
Specifies the trace raw data compression level. Valid values are 0 to 9. A value of zero indicates no compression. A value of 9 provides the highest compression ratio, but at a lower speed. The default is 6.

Note: This option applies only to Linux nodes.

--tracedev-write-mode={blocking | `overwrite`}
Specifies when to overwrite the old data, where:

blocking
Specifies that if the trace buffer is full, wait until the trace data is written to the local disk and the buffer becomes available again to overwrite the old data.

overwrite
Specifies that if the trace buffer is full, overwrite the old data. This is the default.

Note: This option applies only to Linux nodes.

--tracedev-timeformat={`relative` | `absolute` | `calendar`}
Controls time formatting in the trace records. The following values are accepted:

relative
Displays the trace time stamp in relative format, showing the number of seconds from the beginning time stamp. This is the default.

absolute
Displays the trace time stamp in absolute format, showing the number of seconds since 1/1/1970.

calendar
Displays the trace time stamp in local calendar format, showing day of the week, month, day, hours, minutes, seconds, and year.

--tracedev-overwrite-buffer-size=*Size*
Specifies the trace buffer size for Linux trace in overwrite mode. If `--tracedev-write-mode` is set to overwrite, this parameter will be used. It should be no less than 16M. The default is 64M.

Note: This option applies only to Linux nodes.

--format | --noformat
Enables or disables formatting.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies the nodes that will participate in the tracing of the file system. This option supports all defined node classes (with the exception of `mount`). The default value is `all`.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmtracectl` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

To set trace levels to the defined group **def** and start the traces on all nodes when GPFS starts, issue the following command:

```
mmtracectl --set --trace=def --trace-recycle=global
```

The system displays the following output:

```
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
affected nodes.  This is an asynchronous process.
```

To confirm the change, issue the following command:

```
mmlsconfig trace,traceRecycle
```

The system displays output like the following example:

```
trace all 4 tm 2 thread 1 mutex 1 vnode 2 ksvfs 3 klockl 2 io 3 pgalloc 1 mb 1 lock 2 fsck 3
traceRecycle global
```

To manually start traces on all nodes, issue the following command:

```
mmtracectl --start
```

See also

- “mmchconfig command” on page 170

See the `mmtrace` shell script.

Location

`/usr/lpp/mmfs/bin`

mmumount command

Unmounts GPFS file systems on one or more nodes in the cluster.

Synopsis

```
mmumount {Device | MountPoint | DriveLetter |
          all | all_local | all_remote | {-F DeviceFileName}}
         [-f] [-a | -N {Node[,Node...] | NodeFile | NodeClass}]
```

or

```
mmumount Device -f -C {all_remote | ClusterName} [-N Node[,Node...]]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Another name for the `mmumount` command is the `mmunmount` command. Either name can be used.

The `mmumount` command unmounts a previously mounted GPFS file system on one or more nodes in the cluster. If no nodes are specified, the file systems are unmounted only on the node from which the command was issued. The file system can be specified using its device name or the mount point where it is currently mounted.

Use the first form of the command to unmount file systems on nodes that belong to the local cluster.

Use the second form of the command with the `-C` option when it is necessary to force an unmount of file systems that are owned by the local cluster, but are mounted on nodes that belong to another cluster.

When a file system is unmounted by force with the second form of the `mmumount` command, the affected nodes may still show the file system as mounted, but the data will not be accessible. It is the responsibility of the system administrator to clear the mount state by issuing the `umount` command.

When multiple nodes are affected and the unmount target is identified via a mount point or a Windows drive letter, the mount point is resolved on each of the target nodes. Depending on how the file systems were mounted, this may result in different file systems being unmounted on different nodes. When in doubt, always identify the target file system with its device name.

Parameters

***Device* | *MountPoint* | *DriveLetter* | all | all_local | all_remote | {-F *DeviceFileName*}**
Indicates the file system or file systems to be unmounted.

Device
Is the device name of the file system to be unmounted. File system names do not need to be fully qualified. **fs0** is as acceptable as **/dev/fs0**.

MountPoint
Is the location where the GPFS file system to be unmounted is currently mounted.

DriveLetter
Identifies a file system by its Windows drive letter.

all
Indicates all file systems that are known to this cluster.

all_local
Indicates all file systems that are owned by this cluster.

all_remote
Indicates all files systems that are owned by another cluster to which this cluster has access.

-F *DeviceFileName*
Specifies a file containing the device names, one per line, of the file systems to be unmounted.

This must be the first parameter.

Options

-a
Unmounts the file system on all nodes in the GPFS cluster.

-f
Forces the unmount to take place even though the file system may be still in use.

Use this flag with *extreme caution*. Using this flag may cause outstanding write operations to be lost. Because of this, forcing an unmount can cause data integrity failures and should be used with caution.

The `mmumount` command relies on the native `umount` command to carry out the unmount operation. The semantics of forced unmount are platform-specific. On some platforms (such as Linux), even when forced unmount is requested, a file system cannot be unmounted if it is still referenced by the system kernel. Examples of such cases are:

- Open files are present in the file system
- A process uses a subdirectory in the file system as the current working directory
- The file system is NFS-exported

To unmount a file system successfully in such a case, it may be necessary to identify and stop the processes that are referencing the file system. System utilities like `lsof` and `fuser` could be used for this purpose.

-C {all_remote | *ClusterName*}
Specifies the cluster on which the file system is to be unmounted by force. `all_remote` denotes all clusters other than the one from which the command was issued.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies the nodes on which the file system is to be unmounted.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

This command does not support a *NodeClass* of `mount`.

When the `-N` option is specified in conjunction with `-C` *ClusterName*, the specified node names are assumed to refer to nodes that belong to the specified remote cluster (as identified by the `mmlsmount` command). The `mmumount` command cannot verify the accuracy of this information. *NodeClass* and *NodeFile* are not supported in conjunction with the `-C` option.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmumount` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To unmount file system `fs1` on all nodes in the cluster, issue this command:

   ```
   mmumount fs1 -a
   ```

   The system displays output similar to:

   ```
   Fri Feb 10 15:51:25 EST 2006: mmumount: Unmounting file systems ...
   ```

2. To force unmount file system `fs2` on the local node, issue this command:

   ```
   mmumount fs2 -f
   ```

   The system displays output similar to:

   ```
   Fri Feb 10 15:52:20 EST 2006: mmumount: Unmounting file systems ...
   forced unmount of /fs2
   ```

See also

- “mmmount command” on page 545
- “mmlsmount command” on page 517

Location

`/usr/lpp/mmfs/bin`

mmunlinkfileset command

Removes the junction to a GPFS fileset.

Synopsis

```
mmunlinkfileset Device {FilesetName | -J JunctionPath} [-f]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

The `mmunlinkfileset` command removes the junction to the fileset. The junction can be specified by path or by naming the fileset that is its target. The unlink fails if there are files open in the fileset, unless the `-f` flag is specified. The root fileset may not be unlinked.

Attention: If you are using the IBM Spectrum Protect Backup-Archive client, use caution when you unlink filesets that contain data backed up by IBM Spectrum Protect. IBM Spectrum Protect tracks files by pathname and does not track filesets. As a result, when you unlink a fileset, it appears to IBM Spectrum Protect that you deleted the contents of the fileset. Therefore, the IBM Spectrum Protect Backup-Archive client inactivates the data on the IBM Spectrum Protect server which may result in the loss of backup data during the expiration process.

For information on GPFS filesets, see the *IBM Spectrum Scale: Administration Guide*.

Parameters

Device
The device name of the file system that contains the fileset.

File system names need not be fully-qualified. `fs0` is as acceptable as `/dev/fs0`.

FilesetName
Specifies the name of the fileset to be removed.

-J *JunctionPath*
Specifies the name of the junction to be removed.

A junction is a special directory entry that connects a name in a directory of one fileset to the root directory of another fileset.

-f
Forces the unlink to take place even though there may be open files. This option forcibly closes any open files, causing an `errno` of ESTALE on their next use of the file.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `mmunlinkfileset` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. This command indicates the current configuration of filesets for file system gpfs1:

```
mmlsfileset gpfs1
```

The system displays output similar to:

```
Filesets in file system 'gpfs1':
Name                    Status    Path
root                    Linked    /gpfs1
fset1                   Linked    /gpfs1/fset
```

This command unlinks fileset fset1 from file system gpfs1:

```
mmunlinkfileset gpfs1 fset1
```

The system displays output similar to:

```
Fileset 'fset1' unlinked.
```

To confirm the change, issue this command:

```
mmlsfileset gpfs1
```

The system displays output similar to:

```
Filesets in file system 'gpfs1':
Name                    Status    Path
root                    Linked    /gpfs1
fset1                   Unlinked  --
```

2. This command indicates the current configuration of filesets for file system gpfs1:

```
mmlsfileset gpfs1
```

The system displays output similar to:

```
Filesets in file system 'gpfs1':
Name                    Status    Path
root                    Linked    /gpfs1
fset1                   Linked    /gpfs1/fset1
```

This command unlinks junction path /gpfs1/fset1 from file system gpfs1:

```
mmunlinkfileset gpfs1 -J /gpfs1/fset1
```

The system displays output similar to:

```
Fileset 'fset1' unlinked.
```

To confirm the change, issue this command:

```
mmlsfileset gpfs1
```

The system displays output similar to:

```
Filesets in file system 'gpfs1':
Name                    Status    Path
root                    Linked    /gpfs1
fset1                   Unlinked  --
```

See also

- “mmchfileset command” on page 224
- “mmcrfileset command” on page 311
- “mmdelfileset command” on page 369
- “mmlinkfileset command” on page 485
- “mmlsfileset command” on page 501

Location

`/usr/lpp/mmfs/bin`

mmuserauth command

Manages the authentication configuration of file and object access protocols. The configuration allows protocol access methods to authenticate users who need to access data that is stored on the system over these protocols.

Synopsis

```
mmuserauth service create  --data-access-method{file|object}
              --type {ldap|local|ad|nis|userdefined}
              --servers[IP address/hostname]
              {[--pwd-file PasswordFile]--user-name | --enable-anonymous-bind}
              [--base-dn]
              [--enable-server-tls][--enable-ks-ssl]
              [--enable-kerberos][--enable-nfs-kerberos]
              [--user-dn][--group-dn][--netgroup-dn]
              [--netbios-name]  [--domain]
              [--idmap-role{master|subordinate}][--idmap-range][--idmap-range-size]
              [--user-objectclass][--group-objectclass][--user-name-attrib]
              [--user-id-attrib][--user-mail-attrib][--user-filter]
              [ --ks-dns-name][--ks-ext-endpoint]
              [--kerberos-server][--kerberos-realm]
              [--unixmap-domains] [--enable-overlapping-unixmap-ranges] [--ldapmap-domains]
```

Or

```
mmuserauth service list [-Y] [ --data-access-method {file|object|all}]
```

Or

```
mmuserauth service check [ --data-access-method {file|object|all}] [-r|--rectify]
            [-N|--nodes {node-list|cesNodes}][--server-reachability]
```

Or

```
mmuserauth service remove  --data-access-method {file|object|all}[--idmapdelete]
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `mmuserauth` commands to create and manage IBM Spectrum Scale protocol authentication and ID mappings.

Parameters

service
Manages the authentication configuration of file and object access protocols with one of the following actions:

create
Configures authentication for file and object access protocols. The authentication method for file and object access protocols cannot be configured together. The **mmuserauth service create** command needs to be submitted separately for configuring authentication for the file and object access protocols each.

list
Displays the details of the authentication method that is configured for both file and object access protocols.

check
Verifies the authentication method configuration details for file and object access protocols. Validates the connectivity to the configured authentication servers. It also supports corrections to the configuration details on the erroneously configured protocol nodes.

remove
Removes the authentication method configuration of file and object access protocols and ID maps if any.

If you plan to remove both, authentication method configuration and ID maps, remove authentication method configuration followed by the removal of ID maps. That is, at first you need to submit the **mmuserauth service remove** command without the `--idmapdelete` option to remove the authentication method configuration and then submit the same command with the `--idmapdelete` option to remove ID maps.

CAUTION: Deleting the authentication method configuration with the ID maps can lead to irrecoverable loss of access to data. Use this option with proper planning.

--data-access-method *{file|object}*
Specifies the access protocols for which the authentication method needs to be configured. The IBM Spectrum Scale system supports file access protocols such as SMB and NFS along with Object access protocols to access data that is stored on the system.

The file data access method is meant for authorizing the users who access data over SMB and NFS protocols.

--type *{ldap|local|ad|nis|userdefined}*
Specifies the authentication method to be configured for accessing data over file and object access protocols.

`ldap` - Defines an external LDAP server as the authentication server. This authentication type is valid for both file and object access protocols.

`ad` - Defines an external Microsoft Active Directory server as the authentication server. This authentication type is valid for both file and object access protocols.

`local` - Defines an internal database stored on IBM Spectrum Scale protocol nodes for authenticating user accessing data over object access protocol. This authentication type is valid for Object access protocol only.

`nis` - Defines an external NIS server as the authentication server. This authentication type is only valid for NFS file access protocol only. The NIS configuration with an IPv6 address is not supported.

`userdefined` - Defines user-defined (system administrator defined) authentication method for data access. This authentication type is valid for both file and object access protocols.

--servers *[AuthServer1[:Port],AuthServer2[:Port],AuthServer3[:Port] ...]*
Specifies the host name or IP address of the authentication server that is used for file and object access protocols.

This option is only valid with `--type {ldap|ad|nis}`.

With `--type ldap`, the input value format is `"serverName/serverIP:[port]"`.

Specifying the port value is optional. Default port is 389.

For example,

`--servers ldapserver.mydomain.com:1389`.

For file access protocol, multiple LDAP servers can be specified by using a comma as a separator.

For the object access protocol, only one authentication server must be specified. If multiple servers are specified by using a comma, only the first server in the list is considered as the authentication server for configuration.

With `--type ad`, the input value format is `"serverName/serverIP"`.

For example,

`--servers ldapserver.mydomain.com`.

For the file access protocol, only one authentication server must be specified. Specifying multiple servers is invalid. The AD server accepted while configuration is used to fetch details required for validation and configuration of the authentication method. Post successful configuration, each CES node will query DNS to lookout available Domain Controllers serving the AD domain it is joined to. Among the returned list, the node binds with the best available domain controller.

For the object access protocol, only one authentication server must be specified. If multiple servers are specified by using a comma, only the first server in the list is considered as the authentication server.

With `--type nis`, the input value format is `"serverName/serverIP"`.

For example,

`--servers nisserver.mydomain.com`.

Multiple NIS servers can be specified by using a comma separator. At least one of the specified servers must be available and reachable while configuring the authentication method. This is important for the verification of the specified NIS domain, against which the availability of either `passwd.byname` or `netgroup` map is validated.

When you enter an IPv6 address, ensure that the address is enclosed within square brackets to work correctly.

For example,
`--servers [2001:192::e61f:122:feb7:5df5]`

--base-dn ***ldapBase***
Specifies the LDAP base DN of the authentication server. This option is only valid with `--type {ldap|ad}` for `--data-access-method` object and `--type ldap` for `--data-access-method` file.

--enable-anonymous-bind
Specifies whether to enable anonymous binding with authentication server for various validation operations.

This option is only valid with `--type {ldap|ad}` and `--data-access-method {object}`. This option is mutually exclusive with `--user-name` and password combination.

--user-name ***userName***
Specifies the user name to be used to perform operations against the authentication server. This option is only valid with `--type {ldap|ad}` and `--data-access-method {file|object}`.

This option combined with password is mutually exclusive with `--enable-anonymous-bind`. The specified user name must have sufficient permissions to read user and group attributes from the authentication server.

In case of `--type {ad|ldap}` with `--data-access-method` object, the user name must be specified in complete DN format.

In case of `--type ad` with `--data-access-method` file, the specified username is used to join the cluster to AD domain. It results in creating a machine account for the cluster based on the `--netbios-name` specified in the command. After successful configuration, the cluster connects with its machine account, and not the user used during the domain join. So the specified username after domain join has no role to play in communication with the AD domain controller and can be even deleted from the AD server. The cluster can still keep using AD for authentication via the machine account created.

--pwd-file ***PasswordFile***
Specifies the file containing passwords of administrative users for authentication configuration of file and object access protocols. The password file must be saved under `/var/mmfs/ssl/keyServ/tmp` on the node from which you are running the command. If this option is omitted, the command

prompts for a password. The password file is a security-sensitive file and hence, must have the following characteristics:

- It must be a regular file.
- It must be owned by the root user.
- Only the root user must have permission to read or write it.

A password file for file protocol configuration must have the following format:

```
%fileauth:
password=userpassword
```

where:

fileauth
Stanza name for file protocol

password
Specifies the password of *--user-name*.

Note: With `--type ad` for file authentication, the specified password is only required during the domain joining period. After joining the domain, the password of the machine account of the cluster is used for accessing Active Directory.

A password file for object protocol configuration must have the following format:

```
%objectauth:
password=userpassword
ksAdminPwd=ksAdminPwdpassword
ksSwiftPwd=ksSwiftPwdpassword
```

where:

objectauth
Stanza name for object protocol

password
Specifies the password of *--user-name*.

ksAdminPwd
Specifies the Keystone Administrator's password.

ksSwiftPwd
Specifies the Swift service user's password.

Note: Passwords cannot contain any of the following characters: / : \ @ $ { } and space.

The passwords are stored in the associated Keystone and Swift configuration files. You can change these passwords by using the following commands:

- To change the stored AD or LDAP password, issue the following command:

```
# mmobj config change --ccrfile keystone.conf --section ldap --property password --value NewPassword
```

- To change the stored password for the Swift user, issue the following command:

```
# mmobj config change --ccrfile proxy-server.conf --section filter:authtoken --property password
--value NewPassword
```

--enable-server-tls
Specifies whether to enable TLS communication with the authentication server. With `--data-access-method object`, this option is only valid with `--type {ldap|ad}`. With `--data-access-method file`, this option is only valid with `--type {ldap}`.

This option is disabled by default.

For file access protocol configuration, ensure that the CA certificate is placed in the `/var/mmfs/tmp/` directory with the name `ldap_cacert.pem` on the node that the command is to be run.

For object access protocol configuration, ensure that the CA certificate is placed in the /var/mmfs/tmp/ directory with the name object_ldap_cacert.pem on the node that the command is to be run.

--enable-nfs-kerberos
Specifies whether to enable Kerberized logins for users gaining access by using the NFSv3 and NFSv4 file access protocols.

This option is only valid with --type {ad} and --data-access-method {file}.

This option is disabled by default.

Note: Kerberized NFSv3 and NFSv4 access is only supported for users from AD domains that are configured for fetching the UID/GID information from Active Directory (RFC2307 schema attributes). Such AD domain definition is specified by using the --unixmap-domains option.

--user-dn ldapUserDN
Specifies the LDAP group DN. Restricts search of groups within the specified sub-tree. For CIFS access, the value of this parameter is ignored and a search is performed on the baseDN.

This option is only valid with --type {ldap} and --data-access-method {file}. If this parameter is not set, the system uses the value that is set for baseDN as the default value.

--group-dn *ldapGroupDN*
Specifies the LDAP group suffix. Restricts search of groups within a specified sub-tree.

This option is only valid with --type {ldap} and --data-access-method {file}. If this parameter is not set, the system uses the value that is set for baseDN as the default value.

--netgroup-dn *ldapGroupDN*
Specifies the LDAP netgroup suffix. The system searches the netgroups based on this suffix. The value must be specified in complete DN format.

This option is only valid with --type {ldap}and --data-access-method {file}. Default value is baseDN.

--user-objectclass *userObjectClass*
Specifies the object class of user on the authentication server. Only users with specified object class along with other filter are treated as valid users.

If the --data-access-method is object, this option is only valid with --type {ldap|ad}.

If the --data-access-method is file, this option is only valid with --type {ldap}. With --type ldap, the default value is posixAccount and with --type ad the default value is organizationalPerson.

--group-objectclass *groupObjectClass*
Specifies the object class of group on the authentication server. This option is only valid with --type {ldap} and --data-access-method {file}.

--netbios-name *netBiosName*
Specifies the unique identifier of the resources on a network that are running NetBIOS. This option is only valid with --type {ad|ldap} and --data-access-method {file}.

The NetBIOS name is limited to 15 ASCII characters and must not contain any white space or one of the following characters: / : * ? . " ; |

If AD is selected as the authentication method, the NetBIOS name must be selected carefully. If there are name collisions across multiple IBM Spectrum Scale clusters, or between the AD Domain and the NetBIOS name, the configuration does not work properly. Consider the following points while planning for a naming strategy:

- There must not be NetBIOS name collision between two IBM Spectrum Scale clusters that are configured against the same Active Directory server.
- The domain join of the latter machines revokes the join of the former one.
- The NetBIOS name and the domain name must not collide.
- The NetBIOS name and the short name of the Domain Controllers hosting the domain must not collide.

--domain *domainName*
Specifies the name of the NIS domain. This option is only valid with `--type {nis}` and `--data-access-method {file}`.

The NIS domain that is specified must be served by one of the servers specified with `--server`. This option is mandatory when NIS-based authentication is configured for file access.

--idmap-role *{master|subordinate}*
Specifies the ID map role of the IBM Spectrum Scale system. ID map role of a stand-alone or singular system deployment must be selected `"master"`. The value of the ID map role is important in AFM-based deployments.

This option is only valid with `--type {ad}` and `--data-access-method {file}`.

You can use AD with automatic ID mapping to set up two or more storage subsystems in AFM relationship. The two or more systems configured in a master-subordinate relationship provides a means to synchronize the UIDs and GIDs generated for NAS clients on one system with UIDs and GIDs on the other systems. In the AFM relationship, only one system can be configured as master and other systems must be configured as subordinates. The ID map role of master and subordinate systems are the following:

- **Master:** System creates ID maps on its own.
- **Subordinate:** System does not create ID maps on its own. ID maps must be exported from the master to the subordinate.

While using automatic ID mapping, in order to have same ID maps on systems sharing AFM relationship, you need to export the ID mappings from master to subordinate. The NAS file services are inactive on the subordinate system. If you need to export and import ID maps from one system to another, contact the IBM Support Center.

--idmap-range *lowerValue-higherValue*
Specifies the range of values from which the IBM Spectrum Scale UIDs and GIDs are assigned by the system to the Active Directory users and groups. This option is only valid with `--type {ad}` and `--data-access-method {file}`. The default value is `10000000-299999999`. The lower value of the range must be at least 1000. After configuring the IBM Spectrum Scale system with AD authentication, only the higher value can be increased (this essentially increases the number of ranges).

--idmap-range-size *rangeSize*
Specifies the total number of UIDs and GIDs that are assignable per domain. For example, if `--idmap-range` is defined as 10000000-299999999, and range size is defined as 1000000, 290 domains can be mapped, each consisting of 1000000 IDs.

Choose a value for range size that allows for the highest anticipated RID value among all of the anticipated AD users and AD groups in all of the anticipated AD domains. Choose the range size value carefully because range size cannot be changed after the first AD domain is defined on the IBM Spectrum Scale system.

This option is only valid with `--type {ad}` and `--data-access-method {file}`. Default value is `1000000`.

--unixmap-domains unixDomainMap
Specifies the AD domains for which user ID and group ID should be fetched from the AD server. This option is only valid with `--type {ad}` and `--data-access-method {file}`. The unixDomainMap takes value in this format: `DOMAIN1(L1-H1:{win|unix})[;DOMAIN2(L2-H2:{win|unix})[;DOMAIN3(L3-H3:{win|unix})....]]`

DOMAIN
Use DOMAIN to specify an AD domain for which ID mapping services are to be configured. The name of the domain to be specified must be the NetBIOS domain name. The UIDs and GIDs of the users and groups for the specified DOMAIN are read from the UNIX attributes that are populated in the RFC2307 schema extension of AD server. Any users or groups, from this domain, with missing UID/GID attributes are denied access. Use the L-H format to specify the ID range. All the users or groups from DOMAIN that need access to exports need to have their UIDs or GIDs in the specified range.

The specified range should not intersect with:

- The range specified by using the `--idmap-range` option of the command.
- The range specified for other AD DOMAIN for which ID mapping needs to be done from Active Directory (RFC2307 schema attributes) specified in `--unixmap-domains` option.

  However, you can use the `--enable-overlapping-unixmap-ranges` option to allow overlapping ID map ranges for multiple AD domains for which ID mapping needs to be done from Active Directory (RFC2307 schema attributes).
- The range specified for other AD DOMAIN for which ID mapping needs to be done from LDAP server specified in the `--ldapmap-domains` option.

The command reports a failure if you attempt to run the command with such configurations. This is intended to avoid ID collisions among users and groups from different domains.

win: Specifies the system to read the primary group set as Windows primary group of a user on the Active Directory.

unix: Specifies the system to read the primary group as set in "UNIX attributes" of a user on the Active Directory.

For example,

```
--unixmap-domains
"MYDOMAIN1(20000-50000:unix);MYDOMAIN2(100000-200000:win)"
```

--enable-overlapping-unixmap-ranges

Allows overlapping ranges for multiple AD domains that are specified by using `--unixmap-domains` option. This option is only valid with `--data-access-method {file}` and `--type {ad}` along with `--unixmap-domains`.

Note: Ensure that UIDs and GIDs are unique among users and groups from different domains. ID collisions can cause data access issues and compromise data security.

--ldapmap-domains ldapDomainMap

Specifies the AD domains for which user ID and group ID should be fetched from a separate stand-alone LDAP server. This option is only valid with `--type {ad}` and `--data-access-method {file}`. ldapDomainMap takes value of the format as follows,

```
DOMAIN1 (type=stand-alone:ldap_srv=ldapServer:range=Range:usr_dn=userDN:grp_dn=groupDN:[bind_dn=bindDN]
:[bind_dn_pwd=bindDNpassword])[;DOMAIN2(type=stand-alone:ldap_srv=ldapServer:range=Range:usr_dn=userDN
:grp_dn=groupDN:[bind_dn=bindDN]:[bind_dn_pwd=bindDNpassword])[;DOMAIN3(type=stand-
alone:ldap_srv=ldapServer
:range=Range:usr_dn=userDN:grp_dn=groupDN:[bind_dn=bindDN]:[bind_dn_pwd=bindDNpassword])...]]
```

DOMAIN

Use DOMAIN to specify an AD domain for which ID mapping services are to be configured. The name of the domain to be specified must be the Pre-Win2K domain name. The UID and GID of the users and groups for the specified DOMAIN are read from the objects stored on LDAP server in RFC2307 schema attributes. Any users or groups, from this domain, with missing UID/GID attributes are denied access.

type

Defines the type of LDAP server to use.

Supported value: stand-alone.

range

Attribute takes value in the L-H format. Defines the user or group from DOMAIN that needs access to exports need to have their UIDs or GIDs in the specified range. The specified range should not intersect with,

- The range specified using --idmap-range option of the command
- The range specified for other AD DOMAIN for which ID mapping needs to be done from Active Directory (RFC2307 schema attributes) specified in --unixmap-domains option

- The range specified for other AD DOMAIN for which ID mapping needs to be done from LDAP server specified in --ldapmap-domains option

  This is intended to avoid ID collisions among users and groups from different domains.

ldap_srv
Defines the name or IP address of the LDAP server to fetch the UID or GID for of a user or group records in RFC2307 schema format. The user and group objects should be in RFC2307 schema format. Specifying only single LDAP server is supported.

When you enter an IPv6 address, ensure that the address is enclosed within square brackets to work correctly.

For example,
--servers [2001:192::e61f:122:feb7:5df5]

user_dn
Defines the bind tree on the LDAP server where user objects shall be found.

grp_dn
Defines the bind tree on the LDAP server where the group objects shall be found.

bind_dn
Optional attribute.

Defines the user DN that should be used for authentication against the LDAP server. If not specified, anonymous bind shall be performed against the LDAP server.

bind_dn_pwd
Optional attribute.

Defines the password of the user DN specified in bind_dn to be used for authentication against the LDAP server. Must be specified when bind_dn attribute is specified for binding with the LDAP server in the DOMAIN definition.

Password cannot contain these special characters such as semicolon (;) or colon (:).

For example,

```
--ldapmap-domains "MYDOMAIN1(type=stand-alone:range=10000-50000
:ldap_srv=myldapserver.mydomain.com :usr_dn=ou=People,dc=mydomain,dc=com
:grp_dn=ou=Groups,dc=mydomain,dc=com :bind_dn=cn=ldapuser,dc=mydomain,dc=com
:bind_dn_pwd=MYPASSWORD);MYDOMAIN2(type=stand-alone :range=70000-100000
:ldap_srv=myldapserver.example.com:usr_dn=ou=People,dc=example,dc=com
:grp_dn=ou=Groups,dc=example,dc=com)"
```

--enable-kerberos
Specifies whether to enable Kerberized logins for users who are gaining access by using file access protocols.

This option is only valid with `--type {ldap}` and **`--data-access-method {file}`**.

This option is disabled by default.

Note: Ensure that the legitimate keytab file is placed in the `/var/mmfs/tmp` directory and is named as `krb5.keytab` on the node that the authentication method configuration command is to be run.

--kerberos-server *kerberosServer*
Specifies the Kerberos server. This option is only valid with `--type {ldap}` and `--data-access-method {file}`.

When you enter an IPv6 address, ensure that the address is enclosed within square brackets to work correctly.

For example,
--servers [2001:192::e61f:122:feb7:5df5]

--kerberos-realm *kerberosRealm*
Indicates the Kerberos server authentication administrative domain. The realm name is usually the all-uppercase version of the domain name. This option is case-sensitive.

--user-name-attrib *UserNameAttribute*
Specifies the attribute to be used to search for user name on authentication server.

If the `--data-access-method` is `object`, this option is only valid with `--type {ldap|ad}`.

If the `--data-access-method` is `file`, this option is only valid with `--type {ldap}`. With `--type ldap`, default value is `cn` and with `--type ad`, the default value is `sAMAccountName`.

--user-id-attrib *UserIDAttribute*
Specifies the attribute to be used to search for user ID on the authentication server.

If `--data-access-method` is `object`, this option is only valid with `--type {ldap|ad}`.

If `--data-access-method` is `file`, this option is only valid with `--type {ldap}`. For `--type ldap`, default value is `uid` and for `--type ad` the default value is `CN`.

--user-mail-attrib *UserMailAttribute*
Specifies the attribute to be used to search for email on authentication server. If the `--data-access-method` is object, this option is only valid with `--type {ldap|ad}`. For `--data-access-method file`, this option is only valid with `--type {ldap}`. Default value is `mail`.

--user-filter *userFilter*
Specifies the additional filter to be used to search for user in the authentication server. The filter must be specified in LDAP filter format. This option is only valid with `--type {ldap|ad}` and `--data-access-method {object}`. By default, no filter is used.

--ks-dns-name *keystoneDnsName*
Specifies the DNS name for keystone service. The specified name must be resolved on all protocol nodes for proper functioning. This is optional with `--data-access-method {object}`. If the value is not specified for this parameter, the **mmuserauth service create** command uses the value that is used during the IBM Spectrum Scale system installation.

--ks-admin-user *keystoneAdminName*
Specifies the Keystone server administrative user. This user must be a valid user on authentication server if `--type {ldap|ad}` is specified. In case of `--type local`, new user is created, and admin role is assigned in Keystone. This option is mandatory with `--data-access-method {object}`.

For `--type {ldap|ad}`, do not specify user name in DN format for --ks-admin-user. The name must be the base or short name that is written against the specified `user-id-attrib` or `user-name-attrib` of user on the LDAP server.

--enable-ks-ssl
Specifies whether the SSL communication must be enabled with the Keystone service. It enables a secured way to access the Keystone service over the HTTPS protocol. The default communication option with the Keystone service is over HTTP protocol, which has security risks.

This option is only valid with `--data-access-method {object}`.

By default, this option is disabled.

With **--type local | ad | ldap**, ensure that the valid certificate files are placed in the `/var/mmfs/tmp` directory on the node that the command has to be run:

The SSL certificate at: `/var/mmfs/tmp/ssl_cert.pem`

The private key at: `/var/mmfs/tmp/ssl_key.pem`

The CA certificate at: `/var/mmfs/tmp/ssl_cacert.pem`

With `--type userdefined`, ensure that the valid certificate files are placed in the `/var/mmfs/tmp` directory on the node that the command has to be run:

The CA certificate at: `/var/mmfs/tmp/ssl_cacert.pem`

--ks-swift-user *keystoneSwiftName*
Specifies the username to be used as swift user in proxy-server.conf. If AD or LDAP-based authentication is used, this user must be available in the AD or LDAP authentication server. If local authentication method is used, new user with this name is created in the local database This option is only valid with `--data-access-method {object}`.

For `--type {ldap|ad}`, do not specify user name in DN format for --ks-swift-user. The name must be the base or short name that is written against the specified `user-id-attrib` or `user-name-attrib` of user on the LDAP server.

--ks-ext-endpoint *externalendpoint*
Specifies the endpoint URL of external keystone. Only API v3 and HTTP are supported. This option is only valid with `--data-access-method {object}` and `--type {userdefined}`

--idmapdelete
Specifies whether to delete the current ID maps (SID to UID/GID mappings) from the ID mapping databases for the file access method and user-role-project-domain mappings stored in local keystone database for object access method.

This option is only valid with the **mmuserauth service remove** command. Unless the **--data-access-method** parameter is specified on the command line, ID maps for file and object access protocols are erased by default. To delete the ID maps of a particular access protocol, explicitly specify the **--data-access-method** parameter on command line along with the valid access protocol name.

The authentication method configuration and ID maps cannot be deleted together. The authentication method configuration must be deleted before the ID maps.

CAUTION: Deleting ID maps can lead to irrecoverable loss of access to data. Use this option with proper planning.

-N|--nodes*{node-list|cesNodes}*
Verifies the authentication configuration on each node. If the specified node is not protocol node, it is ignored. If protocol node is specified, then the system checks configuration on all protocol nodes. If you do not specify a node, the system checks the configuration of only the current node.

-Y
Displays the command output in a parseable format with a colon (:) as a field delimiter. Each column is described by a header.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

-r|--rectify
Rectifies the authentication configurations and missing SSL and TLS certificates.

--server-reachability
Without this flag, the **mmuserauth service check** command only validates whether the authentication configuration files are consistent across the protocol nodes. Use this flag to ensure if the external authentication server is reachable by each protocol node.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Note: When you reconfigure Object protocol access, messages suggesting that a `duplicate key value violates unique constraint` might appear in the system log. Disregard these messages.

Security

You must have root authority to run the `mmuserauth` command.

The node on which the command is issued must be able to run remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

1. To configure Microsoft Active Directory (AD) based authentication with automatic ID mapping for file access, issue this command:

```
# mmuserauth service create --type ad --data-access-method file --netbios-name
specscale --user-name adUser --idmap-role master --servers myADserver
--idmap-range-size 1000000 --idmap-range 10000000-299999999
```

The system displays output similar to this:

```
File Authentication configuration completed successfully.
```

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays output similar to this:

```
FILE access configuration : AD
PARAMETERS               VALUES
-------------------------------------------------
ENABLE_NFS_KERBEROS      false
SERVERS                  "*"
USER_NAME                specscale$
NETBIOS_NAME             specscale
IDMAP_ROLE               master
IDMAP_RANGE              10000000-299999999
IDMAP_RANGE_SIZE         1000000
UNIXMAP_DOMAINS          none
LDAPMAP_DOMAINS          none

OBJECT access not configured
PARAMETERS               VALUES
-------------------------------------------------
```

2. To configure Microsoft Active Directory (AD) based authentication with RFC2307 ID mapping for file access, issue this command:

```
# mmuserauth service create  --type ad --data-access-method file
--netbios-name specscale --user-name adUser --idmap-role master
--servers myAdserver   --idmap-range-size 1000000
--idmap-range 10000000-299999999 --unixmap-domains 'DOMAIN(5000-20000:win)'
```

The system displays output similar to this:

```
File Authentication configuration completed successfully.
```

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays output similar to this:

```
FILE access configuration : AD
PARAMETERS               VALUES
-------------------------------------------------
ENABLE_NFS_KERBEROS      false
SERVERS                  "*"
USER_NAME                specscale$
NETBIOS_NAME             specscale
IDMAP_ROLE               master
IDMAP_RANGE              10000000-299999999
IDMAP_RANGE_SIZE         1000000
UNIXMAP_DOMAINS          DOMAIN(5000-20000:win)
LDAPMAP_DOMAINS          none
```

```
OBJECT access not configured
PARAMETERS              VALUES
```

3. To configure Microsoft Active Directory (AD) based authentication with LDAP ID mapping for file access, issue this command:

```
mmuserauth service create --data-access-method file --type ad --servers myADserver
--user-name adUser  --netbios-name specscale --idmap-role master
--ldapmap-domains "DOMAIN(type=stand-alone: range=1000-10000:ldap_srv=myLDAPserver:
usr_dn=ou=People,dc=example,dc=com:grp_dn=ou=Groups,dc=example,
dc=com:bind_dn=cn=ldapuser,dc=example,dc=com:bind_dn_pwd=password)"
```

The system displays output similar to this:

```
File Authentication configuration completed successfully.
```

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays output similar to this:

```
FILE access configuration : AD
PARAMETERS              VALUES
-------------------------------------------------
ENABLE_NFS_KERBEROS     false
SERVERS                 "*"
USER_NAME               specscale$
NETBIOS_NAME            specscale
IDMAP_ROLE              master
IDMAP_RANGE             10000000-299999999
IDMAP_RANGE_SIZE        1000000
UNIXMAP_DOMAINS         none
LDAPMAP_DOMAINS         DOMAIN(type=stand-alone: range=1000-10000:
ldap_srv=myLDAPserver:usr_dn=ou=People,dc=example,dc=com:grp_dn=
ou=Groups,dc=example,dc=com:bind_dn=cn=ldapuser,dc=example,dc=com)

OBJECT access not configured
PARAMETERS              VALUES
-------------------------------------------------
```

4. To configure Microsoft Active Directory (AD) based authentication with LDAP ID mapping for file access (anonymous binding with LDAP), issue this command:

```
# mmuserauth service create --data-access-method file --type ad
--servers myADserver --user-name adUser
--netbios-name specscale --idmap-role master --ldapmap-domains
"DOMAIN(type=stand-alone: range=1000-10000:ldap_srv=myLDAPserver:
usr_dn=ou=People,dc=example,dc=com:grp_dn=ou=Groups,dc=example,dc=com)"
```

The system displays output similar to this:

```
File Authentication configuration completed successfully.
```

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays output similar to this:

```
FILE access configuration : AD
PARAMETERS              VALUES
-------------------------------------------------
ENABLE_NFS_KERBEROS     false
SERVERS                 "*"
USER_NAME               specscale$
NETBIOS_NAME            specscale
```

```
IDMAP_ROLE              master
IDMAP_RANGE             10000000-299999999
IDMAP_RANGE_SIZE        1000000
UNIXMAP_DOMAINS         none
LDAPMAP_DOMAINS         DOMAIN(type=stand-alone: range=1000-10000:ldap_srv=myLDAPserver:
usr_dn=ou=People,dc=example,dc=com:grp_dn=ou=Groups,dc=example,dc=com)

OBJECT access not configured
PARAMETERS              VALUES
-------------------------------------------------
```

5. To configure AD-based authentication with overlapping ID map ranges, issue this command:

```
# mmuserauth service create --data-access-method file --type ad --servers myADserver --user-
name adUser
 --netbios-name specscale --idmap-role master --unixmap-domains "DOMAIN1(2000-4000);
DOMAIN2(2000-4000)"
--enable-overlapping-unixmap-ranges
```

The system displays output similar to this:

```
Enter Active Directory User 'adUser' password:
Enabling Overlapping unixmap ranges. Make sure that UIDs and GIDs are unique in order to
avoid ACLs
or/and data access issues. See man mmuserauth for further details.

File authentication configuration completed successfully.
```

To verify the authentication configuration, the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays output similar to this:

```
# mmuserauth service list
FILE access configuration : AD
PARAMETERS              VALUES
-------------------------------------------------
ENABLE_NFS_KERBEROS     false
SERVERS                 "*"
USER_NAME               specscale$
NETBIOS_NAME            specscale
IDMAP_ROLE              master
IDMAP_RANGE             10000000-299999999
IDMAP_RANGE_SIZE        1000000
UNIXMAP_DOMAINS         DOMAIN1(2000-4000:win);DOMAIN2(2000-4000:win)
LDAPMAP_DOMAINS         none

OBJECT access not configured
PARAMETERS              VALUES
-------------------------------------------------
```

6. To configure LDAP-based authentication with TLS encryption for file access, issue this command:

```
# mmuserauth service create --type ldap --data-access-method file
--servers myLDAPserver --base-dn dc=example,dc=com
--user-name cn=ldapuser,dc=example,dc=com
--netbios-name specscale --enable-server-tls
```

The system displays output similar to this:

```

File Authentication configuration completed successfully.
```

Note: Before issuing the **mmuserauth service create** command to configure LDAP with TLS, ensure that the CA certificate for LDAP server is placed under `/var/mmfs/tmp` directory with the name "ldap_cacert.pem" specifically on the protocol node where the command is issued.

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays output similar to this:

```
FILE access configuration : LDAP
PARAMETERS               VALUES
-------------------------------------------------
ENABLE_ANONYMOUS_BIND    false
ENABLE_SERVER_TLS        true
ENABLE_KERBEROS          false
USER_NAME                cn=ldapuser,dc=example,dc=com
SERVERS                  myLDAPserver
NETBIOS_NAME             specscale
BASE_DN                  dc=example,dc=com
USER_DN                  none
GROUP_DN                 none
NETGROUP_DN              none
USER_OBJECTCLASS         posixAccount
GROUP_OBJECTCLASS        posixGroup
USER_NAME_ATTRIB         cn
USER_ID_ATTRIB           uid
KERBEROS_SERVER          none
KERBEROS_REALM           none

OBJECT access not configured
PARAMETERS               VALUES
-------------------------------------------------
```

7. To configure LDAP-based authentication with Kerberos for file access, issue this command:

```
# mmuserauth service create --type ldap --data-access-method file
--servers myLDAPserver --base-dn dc=example,dc=com
--user-name cn=ldapuser,dc=example,dc=com
--netbios-name specscale --enable-kerberos
--kerberos-server myKerberosServer  --kerberos-realm MYREAL.com
```

The system displays output similar to this:

```
File Authentication configuration completed successfully.
```

Note: Before issuing the **mmuserauth service create** command to configure LDAP with Kerberos, ensure that the keytab file is also placed under `/var/mmfs/tmp` directory name as "krb5.keytab" specifically on the node where the command is run.

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays output similar to this:

```
FILE access configuration : LDAP
PARAMETERS               VALUES
-------------------------------------------------
ENABLE_ANONYMOUS_BIND    false
ENABLE_SERVER_TLS        false
ENABLE_KERBEROS          true
USER_NAME                cn=ldapuser,dc=example,dc=com
SERVERS                  myLDAPserver
NETBIOS_NAME             specscale
BASE_DN                  dc=example,dc=com
USER_DN                  none
GROUP_DN                 none
NETGROUP_DN              none
USER_OBJECTCLASS         posixAccount
GROUP_OBJECTCLASS        posixGroup
USER_NAME_ATTRIB         cn
USER_ID_ATTRIB           uid
```

```
KERBEROS_SERVER          myKerberosServer
KERBEROS_REALM           MYREAL.com

OBJECT access not configured
PARAMETERS               VALUES
-------------------------------------------------
```

8. To configure LDAP with TLS and Kerberos for file access, issue this command:

```
# mmuserauth service create --type ldap --data-access-method file
--servers myLDAPserver --base-dn dc=example,dc=com
--user-name cn=ldapuser,dc=example,dc=com
--netbios-name specscale --enable-server-tls --enable-kerberos
      --kerberos-server myKerberosServer --kerberos-realm MYREAL.com
```

The system displays output similar to this:

```
File Authentication configuration completed successfully.
```

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays output similar to this:

```
FILE access configuration : LDAP
PARAMETERS               VALUES
-------------------------------------------------
ENABLE_ANONYMOUS_BIND    false
ENABLE_SERVER_TLS        true
ENABLE_KERBEROS          true
USER_NAME                cn=ldapuser,dc=example,dc=com
SERVERS                  myLDAPserver
NETBIOS_NAME             specscale
BASE_DN                  dc=example,dc=com
USER_DN                  none
GROUP_DN                 none
NETGROUP_DN              none
USER_OBJECTCLASS         posixAccount
GROUP_OBJECTCLASS        posixGroup
USER_NAME_ATTRIB         cn
USER_ID_ATTRIB           uid
KERBEROS_SERVER          myKerberosServer
KERBEROS_REALM           MYREAL.com

OBJECT access not configured
PARAMETERS               VALUES
-------------------------------------------------
```

9. To configure LDAP without TLS and without Kerberos for file access, issue this command:

```
# mmuserauth service create --type ldap --data-access-method file
--servers myLDAPserver --base-dn dc=example,dc=com --user-name
cn=ldapuser,dc=example,dc=com  --netbios-name specscale
```

The system displays output similar to this:

```
File Authentication configuration completed successfully.
```

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays output similar to this:

```
FILE access configuration : LDAP
PARAMETERS               VALUES
-------------------------------------------------
ENABLE_ANONYMOUS_BIND    false
```

```
ENABLE_SERVER_TLS        false
ENABLE_KERBEROS          false
USER_NAME                cn=ldapuser,dc=example,dc=com
SERVERS                  myLDAPserver
NETBIOS_NAME             specscale
BASE_DN                  dc=example,dc=com
USER_DN                  none
GROUP_DN                 none
NETGROUP_DN              none
USER_OBJECTCLASS         posixAccount
GROUP_OBJECTCLASS        posixGroup
USER_NAME_ATTRIB         cn
USER_ID_ATTRIB           uid
KERBEROS_SERVER          none
KERBEROS_REALM           none

OBJECT access not configured
PARAMETERS               VALUES
-------------------------------------------------
```

10. To configure NIS-based authentication for file access, issue this command:

```
# mmuserauth service create --type nis --data-access-method file
--servers myNISserver --domain nisdomain
```

The system displays output similar to this:

```
File Authentication configuration completed successfully.
```

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays output similar to this:

```
FILE access configuration : NIS
PARAMETERS               VALUES
-------------------------------------------------
SERVERS                  myNISserver
DOMAIN                   nisdomain

OBJECT access not configured
PARAMETERS               VALUES
-------------------------------------------------
```

11. To configure user-defined authentication for file access, issue this command:

```
# mmuserauth service create --data-access-method file --type userdefined
```

The system displays output similar to this:

```
File Authentication configuration completed successfully.
```

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays output similar to this:

```
FILE access configuration : USERDEFINED
PARAMETERS               VALUES
-------------------------------------------------
OBJECT access not configured
PARAMETERS               VALUES
-------------------------------------------------
```

12. To configure local authentication for object access, issue this command:

```
# mmuserauth service create --data-access-method object --type local
--ks-dns-name ksDNSname --ks-admin-user admin
```

The system displays output similar to this:

```
Object configuration with local (Database) as identity backend is completed
successfully.
Object Authentication configuration completed successfully.
```

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays output similar to this:

```
FILE access not configured
PARAMETERS               VALUES
-------------------------------------------------

OBJECT access configuration : LOCAL
PARAMETERS               VALUES
-------------------------------------------------
ENABLE_KS_SSL            false
ENABLE_KS_CASIGNING      false
KS_ADMIN_USER            admin
```

13. To configure AD without TLS authentication for object access, issue this command:

```
# mmuserauth service create --type ad --data-access-method object
--user-name "cn=adUser,cn=Users,dc=example,dc=com"  --base-dn "dc=example,DC=com"
--ks-dns-name ksDNSname --ks-admin-user admin --servers myADserver --user-id-attrib cn
--user-name-attrib sAMAccountName --user-objectclass organizationalPerson --user-dn
"cn=Users,dc=example,dc=com"
--ks-swift-user swift
```

The system displays output similar to this:

```
Object configuration with LDAP (Active Directory) as identity backend is completed
successfully.
Object Authentication configuration completed successfully.
```

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays output similar to this:

```
FILE access not configured
PARAMETERS               VALUES
-------------------------------------------------

OBJECT access configuration: AD
PARAMETERS               VALUES
-------------------------------------------------
ENABLE_ANONYMOUS_BIND    false
ENABLE_SERVER_TLS        false
ENABLE_KS_SSL            false
USER_NAME                cn=adUser,cn=Users,dc=example,dc=com
SERVERS                  myADserver
BASE_DN                  dc=IBM,DC=local
USER_DN                  cn=users,dc=example,dc=com
USER_OBJECTCLASS         organizationalPerson
USER_NAME_ATTRIB         sAMAccountName
USER_ID_ATTRIB           cn
USER_MAIL_ATTRIB         mail
USER_FILTER              none
ENABLE_KS_CASIGNING      false
KS_ADMIN_USER            admin
```

14. To configure AD with TLS authentication for object access, issue this command:

```
# mmuserauth service create --type ad --data-access-method object
--user-name "cn=adUser,cn=Users,dc=example,dc=com"  --base-dn
"dc=example,DC=com" --enable-server-tls --ks-dns-name ksDNSname --ks-admin-user admin --servers
myADserver --user-id-attrib cn --user-name-attrib sAMAccountName --user-objectclass organizationalPerson
--user-dn "cn=Users,dc=example,dc=com" --ks-swift-user swift
```

The system displays output similar to this:

```
Object configuration with LDAP (Active Directory) as identity backend is completed
successfully.
Object Authentication configuration completed successfully.
```

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays output similar to this:

```
FILE access not configured
PARAMETERS               VALUES
-------------------------------------------------
OBJECT access configuration: AD
PARAMETERS               VALUES
-------------------------------------------------
ENABLE_ANONYMOUS_BIND    false
ENABLE_SERVER_TLS        true
ENABLE_KS_SSL            false
USER_NAME                cn=adUser,cn=Users,dc=example,dc=com
SERVERS                  myADserver
BASE_DN                  dc=IBM,DC=com
USER_DN                  cn=users,dc=example,dc=com
USER_OBJECTCLASS         organizationalPerson
USER_NAME_ATTRIB         sAMAccountName
USER_ID_ATTRIB           cn
USER_MAIL_ATTRIB         mail
USER_FILTER              none
ENABLE_KS_CASIGNING      false
KS_ADMIN_USER            admin
```

15. To configure LDAP-based authentication for object access, issue this command:

```
# mmuserauth service create --type ldap --data-access-method object
--user-name "cn=ldapuser,dc=example,dc=com"
--base-dn dc=example,dc=com --ks-dns-name ksDNSname --ks-admin-user admin
--servers myLDAPserver --user-dn "ou=People,dc=example,dc=com"
--ks-swift-user swift
```

The system displays output similar to this:

```
Object configuration with LDAP as identity backend is completed successfully.
Object Authentication configuration completed successfully.
```

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays output similar to this:

```
FILE access not configured
PARAMETERS               VALUES
-------------------------------------------------
OBJECT access configuration : LDAP
PARAMETERS               VALUES
-------------------------------------------------
ENABLE_SERVER_TLS        false
ENABLE_KS_SSL            false
USER_NAME                cn=ldapuser,dc=example,dc=com
SERVERS                  myLDAPserver
BASE_DN                  dc=example,dc=com
USER_DN                  ou=people,dc=example,dc=com
```

```
USER_OBJECTCLASS         posixAccount
USER_NAME_ATTRIB         cn
USER_ID_ATTRIB           uid
USER_MAIL_ATTRIB         mail
USER_FILTER              none
ENABLE_KS_CASIGNING      false
KS_ADMIN_USER            admin
```

16. To configure LDAP with TLS-based authentication for object access, issue this command:

```
# mmuserauth service create --type ldap --data-access-method object
--user-name "cn=ldapuser,dc=example,dc=com"
--base-dn dc=example,dc=com --enable-server-tls
--ks-dns-name ksDNSname --ks-admin-user admin --servers myLDAPserver
--user-dn "ou=People,dc=example,dc=com" --ks-swift-user swift
```

The system displays output similar to this:

```
Object configuration with LDAP as identity backend is completed successfully.
Object Authentication configuration completed successfully.
```

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays output similar to this:

```
FILE access not configured
PARAMETERS               VALUES
-------------------------------------------------

OBJECT access configuration : LDAP
PARAMETERS               VALUES
-------------------------------------------------
ENABLE_SERVER_TLS        true
ENABLE_KS_SSL            false
USER_NAME                cn=ldapuser,dc=example,dc=com
SERVERS                  myLDAPserver
BASE_DN                  dc=example,dc=com
USER_DN                  ou=people,dc=example,dc=com
USER_OBJECTCLASS         posixAccount
USER_NAME_ATTRIB         cn
USER_ID_ATTRIB           uid
USER_MAIL_ATTRIB         mail
USER_FILTER              none
ENABLE_KS_CASIGNING      false
KS_ADMIN_USER            admin
```

17. To remove the authentication method that is configured for file access, issue this command:

```
# mmuserauth service remove --data-access-method file
```

The system displays output similar to this:

```
mmuserauth service remove: Command successfully completed
```

Note: Authentication configuration and ID maps cannot be deleted together. To remove ID maps, remove the authentication configuration first and then remove ID maps. Also, you cannot delete ID maps that are used for file and object access together. That is, when you delete the ID maps, the value that is specified for `--data-access-method` must be either `file` or `object`.

18. To remove the authentication method that is configured for object access, issue this command:

```
# mmuserauth service remove --data-access-method object
```

The system displays output similar to this:

```
mmuserauth service remove: Command successfully completed
```

Note: Authentication configuration and ID maps cannot be deleted together. To remove ID maps, remove the authentication configuration first and then remove the ID maps. Also, you cannot delete ID maps that are used for file and object access together. That is, when you delete the ID maps, the value that is specified for `--data-access-method` must be either `file` or `object`.

19. To check whether the authentication configuration is consistent across the cluster and the required services are enabled and running, issue this command:

```
mmuserauth service check --data-access-method file --nodes cesNodes  --server-reachability
```

The system displays output similar to this:

```
        Userauth file check on node: node1
                  Checking SSSD_CONF: OK
                  Checking nsswitch file: OK
                  Checking Pre-requisite Packages: OK
        LDAP servers status
                  LDAP server myLdapServer : OK
        Service 'sssd' status: OK

        Userauth file check on node: node2
                  Checking SSSD_CONF: OK
                  Checking nsswitch file: OK
                  Checking Pre-requisite Packages: OK
        LDAP servers status
                  LDAP server myLdapServer : OK
        Service 'sssd' status: OK
```

20. To check whether the file authentication configuration is consistent across the cluster and the required services are enabled and running, and if you want to correct the situation, issue this command:

```
mmuserauth service check --data-access-method file --nodes cesNodes  --rectify
```

21. To check that all object configuration files (including certificates) are present, and if not, rectify the situation by issuing the following command:

```
# mmuserauth service check --data-access-method object --rectify
```

The system displays output similar to this:

```
Userauth object check on node: node1
Checking keystone.conf: OK
Checking /etc/keystone/ssl/certs/signing_cert.pem: OK
Checking /etc/keystone/ssl/private/signing_key.pem: OK
Checking /etc/keystone/ssl/certs/signing_cacert.pem: OK
Checking /etc/keystone/ssl/certs/object_ldap_cacert.pem: OK
Service 'openstack-keystone' status: OK
```

22. To check if the external authentication server is reachable by each protocol node, use the following command:

```
mmuserauth service check --server-reachability
```

a. If file is not configured, object is configured, and there are no errors, the system displays output similar to this:

```
Userauth object check on node: node1
    Checking keystone.conf: OK
    Checking wsgi-keystone.conf: OK
    Checking /etc/keystone/ssl/certs/signing_cert.pem: OK
    Checking /etc/keystone/ssl/private/signing_key.pem: OK
    Checking /etc/keystone/ssl/certs/signing_cacert.pem: OK

LDAP servers status
    LDAP server myLDAPserver : OK
Service 'httpd' status: OK
```

b. If file is not configured, object is configured, and there is a single error, the system displays output similar to this:

```
Userauth object check on node: node1
    Checking keystone.conf: OK
    Checking wsgi-keystone.conf: OK
    Checking /etc/keystone/ssl/certs/signing_cert.pem: OK
    Checking /etc/keystone/ssl/private/signing_key.pem: OK
    Checking /etc/keystone/ssl/certs/signing_cacert.pem: OK

LDAP servers status
    LDAP server myLDAPserver : ERROR
Service 'httpd' status: OK
```

c. If file and object are configured and there are no errors, the system displays output similar to this:

```
Userauth file check on node: node1
    Checking nsswitch file: OK
       Checking Pre-requisite Packages: OK
       Checking SRV Records lookup: OK

Domain Controller status
    NETLOGON connection: OK, connection to DC: win2k16.example.com
    Domain join status: OK
    Machine password status: OK
Service 'gpfs-winbind' status: OK

Userauth object check on node: node1
    Checking keystone.conf: OK
    Checking wsgi-keystone.conf: OK
    Checking /etc/keystone/ssl/certs/signing_cert.pem: OK
    Checking /etc/keystone/ssl/private/signing_key.pem: OK
    Checking /etc/keystone/ssl/certs/signing_cacert.pem: OK

LDAP servers status
    LDAP server myLDAPserver : OK
Service 'httpd' status: OK
```

d. If file and object are configured and there is a single error, the system displays output similar to this:

```


Userauth file check on node: node1
    Checking nsswitch file: OK
       Checking Pre-requisite Packages: OK
       Checking SRV Records lookup: OK

Domain Controller status
    NETLOGON connection: OK, connection to DC: WIN8-12Up.example.com
    Domain join status: OK
    Machine password status: ERROR
Service 'gpfs-winbind' status: OK

Userauth object check on node: node1


    Checking keystone.conf: OK
    Checking wsgi-keystone.conf: OK
    Checking /etc/keystone/ssl/certs/signing_cert.pem: OK
    Checking /etc/keystone/ssl/private/signing_key.pem: OK
    Checking /etc/keystone/ssl/certs/signing_cacert.pem: OK

LDAP servers status
    LDAP server myLDAPserver : OK
```

e. If file and object are configured and there is are multiple errors, the system displays output similar to this:

```
Userauth file check on node: node1
    Checking nsswitch file: OK
       Checking Pre-requisite Packages: OK
       Checking SRV Records lookup: OK

Domain Controller status
    NETLOGON connection: ERROR (DC not found)
    Domain join status: ERROR
    Machine password status: ERROR
Service 'gpfs-winbind' status: OK
```

```
Userauth object check on node: node1
    Checking keystone.conf: OK
    Checking wsgi-keystone.conf: OK
    Checking /etc/keystone/ssl/certs/signing_cert.pem: OK
    Checking /etc/keystone/ssl/private/signing_key.pem: OK
    Checking /etc/keystone/ssl/certs/signing_cacert.pem: OK

LDAP servers status
    LDAP server myLDAPserver : ERROR
Service 'httpd' status: OK
```

Note: The --rectify or -r option cannot fix server reachability errors. Specifying that option with --server-reachability may fix the erroneous config files and service-related errors only.

23. To configure AD authentication by using a password file for File protocol configuration, use the following command:

```
mmuserauth service create  --type ad --data-access-method file
 --netbios-name test --user-name administrator
--idmap-role master --servers myADServer
 --pwd-file fileauth
```

Contents of `fileauth` saved at `/var/mmfs/ssl/keyServ/tmp/` are:

```
%fileauth:
password=Passw0rd
```

Here, *Passw0rd* is the password for the user to bind with the authentication server.

24. To configure AD authentication by using a password file for Object protocol configuration, use the following command:

```
mmuserauth service create --type ad --data-access-method object
 --base-dn "dc=example,DC=com"  --servers myADserver --user-id-attrib cn
--user-name-attrib sAMAccountName --user-objectclass organizationalPerson
--user-dn "cn=Users,dc=example,dc=com"  --pwd-file objectauth
```

Contents of `objectauth` saved at `/var/mmfs/ssl/keyServ/tmp/` are:

```
%objectauth:
password=Passw0rd
ksAdminPwd=Passw0rd1
ksSwiftPwd=Passw0rd2
```

Here, *Passw0rd* is the password for the user to bind with the authentication server. *Passw0rd1* is the Keystone administrator's password, and *Passw0rd2* is the Swift Service user's password.

25. To check whether the DNS configuration is correct when cluster is already configured with file AD authentication scheme, issue this command:

```
mmuserauth service check --data-access-method file --nodes cesNodes
```

If DNS configuration is valid; then system displays output similar to this:

```
    Userauth file check on node: node1
                  Checking nsswitch file: OK
                  Checking Pre-requisite Packages: OK
                  Checking SRV Records lookup: OK
    Service 'gpfs-winbind' status: OK

    Userauth file check on node: node2
                  Checking nsswitch file: OK
                  Checking Pre-requisite Packages: OK
                  Checking SRV Records lookup: OK
    Service 'gpfs-winbind' status: OK
```

If DNS configuration is incorrect; then system displays output similar to this:

```
   Userauth file check on node: node1
               Checking nsswitch file: OK
               Checking Pre-requisite Packages: OK
               Checking SRV Records lookup: ERROR (Make sure Domain Controller
```

```
   can be looked up from this node. Validate correct DNS server is populated in network
configuration.)
   Found errors in configuration.

   Userauth file check on node: node2
               Checking nsswitch file: OK
               Checking Pre-requisite Packages: OK
               Checking SRV Records lookup: ERROR (Make sure Domain Controller
   can be looked up from this node. Validate correct DNS server is populated in network
configuration.)
        Found errors in configuration.
```

26. To configure Microsoft Active Directory(AD)-based authentication when the time difference between the node and domain controller is more than five minutes, run the following command:

```
mmuserauth service create --type ad --data-access-method file --netbios-name
   specscale --user-name adUser --idmap-role master --servers myADserver
   --idmap-range-size 1000000 --idmap-range 10000000-299999999
```

The system displays output similar to this:

```
   WARNING: Time difference between current node and domain controller is 4073 seconds.
   It is greater than max allowed clock skew 300 seconds.   File Authentication configuration
   completed successfully.
```

27. To configure LDAP-based authentication for file access with an IPv6 address of the authentication server, issue this command:

```
# mmuserauth service create --type ldap --data-access-method file --servers
[2001:192::e61f:122:feb7:5df0] --base-dn dc=example,dc=com --user-name
cn=ldapuser,dc=example,dc=com --netbios-name specscale
```

The system displays output similar to this:

```
File Authentication configuration completed successfully.
```

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays output similar to this:

```
FILE access configuration : LDAP
    PARAMETERS              VALUES
    -------------------------------------------------
    ENABLE_SERVER_TLS       false
    ENABLE_KERBEROS         false
    USER_NAME               cn=ldapuser,dc=example,dc=com
    SERVERS                 [2001:192::e61f:122:feb7:5df0]
    NETBIOS_NAME            specscale
    BASE_DN                 dc=example,dc=com
    USER_DN                 none
    GROUP_DN                none
    NETGROUP_DN             none
    USER_OBJECTCLASS        posixAccount
    GROUP_OBJECTCLASS       posixGroup
    USER_NAME_ATTRIB        cn
    USER_ID_ATTRIB          uid
    KERBEROS_SERVER         none
    KERBEROS_REALM          none

    OBJECT access not configured
    PARAMETERS              VALUES
    -------------------------------------------------
```

28. To configure LDAP-based authentication with TLS encryption for file access with an IPv6 address of the authentication server, issue this command:

```
mmuserauth service create --type ldap --data-access-method file --servers
[2001:192::e61f:122:feb7:5df0] --base-dn dc=example,dc=com --user-name
cn=ldapuser,dc=example,dc=com --netbios-name specscale --enable-server-tls
```

The system displays output similar to this:

```
File Authentication configuration completed successfully.
```

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays output similar to this:

```
FILE access configuration : LDAP
    PARAMETERS              VALUES
    -------------------------------------------------
    ENABLE_SERVER_TLS       true
    ENABLE_KERBEROS         false
    USER_NAME               cn=ldapuser,dc=example,dc=com
    SERVERS                 [2001:192::e61f:122:feb7:5df0]
    NETBIOS_NAME            specscale
    BASE_DN                 dc=example,dc=com
    USER_DN                 none
    GROUP_DN                none
    NETGROUP_DN             none
    USER_OBJECTCLASS        posixAccount
    GROUP_OBJECTCLASS       posixGroup
    USER_NAME_ATTRIB        cn
    USER_ID_ATTRIB          uid
    KERBEROS_SERVER         none
    KERBEROS_REALM          none

    OBJECT access not configured
    PARAMETERS              VALUES
    -------------------------------------------------
```

29. To configure LDAP-based authentication with Kerberos for file access with an IPv6 address of the authentication server, issue this command:

```
mmuserauth service create --type ldap --data-access-method file --servers
[2001:192::e61f:122:feb7:5df0] --base-dn dc=example,dc=com --user-name
cn=ldapuser,dc=example,dc=com --netbios-name specscale --enable-kerberos --kerberos-server
[2001:192::e61f:122:feb7:5dc0] --kerberos-realm MYREALM.com
```

The system displays output similar to this:

```
File Authentication configuration completed successfully.
```

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays output similar to this:

```
FILE access configuration : LDAP
    PARAMETERS              VALUES
    -------------------------------------------------
    ENABLE_SERVER_TLS       false
    ENABLE_KERBEROS         true
    USER_NAME               cn=ldapuser,dc=example,dc=com
    SERVERS                 [2001:192::e61f:122:feb7:5df0]
    NETBIOS_NAME            specscale
    BASE_DN                 dc=example,dc=com
    USER_DN                 none
    GROUP_DN                none
    NETGROUP_DN             none
    USER_OBJECTCLASS        posixAccount
    GROUP_OBJECTCLASS       posixGroup
    USER_NAME_ATTRIB        cn
    USER_ID_ATTRIB          uid
    KERBEROS_SERVER         [2001:192::e61f:122:feb7:5dc0]
    KERBEROS_REALM          MYREALM.com

    OBJECT access not configured
```

```
    PARAMETERS               VALUES
    -------------------------------------------------
```

30. To configure Microsoft Active Directory (AD)-based authentication with the automatic ID mapping for file access with an IPv6 address of the authentication server, issue this command:

```
mmuserauth service create --type ad --data-access-method file --servers
[2001:192::e61f:122:feb7:5df0] --netbios-name specscale --user-name
adUser --idmap-role master --idmap-range-size 1000000 --idmap-range 10000000-299999999
```

The system displays output similar to this:

```
File Authentication configuration completed successfully.
```

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays output similar to this:

```
FILE access configuration : AD
    PARAMETERS               VALUES
    -------------------------------------------------
    ENABLE_NFS_KERBEROS      false
    SERVERS                  "*"
    USER_NAME                adUser$
    NETBIOS_NAME             specscale
    IDMAP_ROLE               master
    IDMAP_RANGE              10000000-299999999
    IDMAP_RANGE_SIZE         1000000
    UNIXMAP_DOMAINS          none
    LDAPMAP_DOMAINS          none

    OBJECT access not configured
    PARAMETERS               VALUES
    -------------------------------------------------
```

31. To configure Microsoft Active Directory (AD)-based authentication with RFC2307 ID mapping for file access with IPv6 address of the authentication server, issue this command:

```
mmuserauth service create --type ad --data-access-method file --servers
[2001:192::e61f:122:feb7:5df0] --netbios-name specscale --user-name
adUser --idmap-role master --unixmap-domains 'TESTDOMAIN(10000-50000:win)'
```

The system displays output similar to this:

```
File Authentication configuration completed successfully.
```

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays output similar to this:

```
FILE access configuration : AD
    PARAMETERS               VALUES
    -------------------------------------------------
    ENABLE_NFS_KERBEROS      false
    SERVERS                  "*"
    USER_NAME                adUser$
    NETBIOS_NAME             specscale
    IDMAP_ROLE               master
    IDMAP_RANGE              10000000-299999999
    IDMAP_RANGE_SIZE         1000000
    UNIXMAP_DOMAINS          TESTDOMAIN(10000-50000:win)
    LDAPMAP_DOMAINS          none

    OBJECT access not configured
```

```
    PARAMETERS               VALUES
    -------------------------------------------------
```

32. To configure Microsoft Active Directory (AD)-based authentication with LDAP ID mapping for file access with IPv6 address of the authentication server, issue this command:

```
mmuserauth service create --type ad --data-access-method file --servers
[2001:192::e61f:122:feb7:5df0] --netbios-name specscale --user-name
adUser --idmap-role master --ldapmap-domains "TESTDOMAIN(type=stand-alone:
range=1000-10000:ldap_srv=[2001:192::e61f:122:feb7:5bf0]:
usr_dn=ou=People,dc=example,dc=com:grp_dn=ou=Groups,dc=example,
dc=com:bind_dn=cn=ldapuser,dc=example,dc=com:bind_dn_pwd=password)"
```

The system displays output similar to this:

```
File Authentication configuration completed successfully.
```

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays output similar to this:

```
FILE access configuration : AD
    PARAMETERS               VALUES
    -------------------------------------------------
    ENABLE_NFS_KERBEROS      false
    SERVERS                  "*"
    USER_NAME                adUser$
    NETBIOS_NAME             specscale
    IDMAP_ROLE               master
    IDMAP_RANGE              10000000-299999999
    IDMAP_RANGE_SIZE         1000000
    UNIXMAP_DOMAINS          none
    LDAPMAP_DOMAINS          TESTDOMAIN(type=stand-alone: range=1000-10000:
    ldap_srv=[2001:192::e61f:122:feb7:5bf0]:usr_dn=ou=People,dc=example,dc=com:grp_dn=
    ou=Groups,dc=example,dc=com:bind_dn=cn=ldapuser,dc=example,dc=com)

    OBJECT access not configured
    PARAMETERS               VALUES
    -------------------------------------------------
```

See also

- “mmces command” on page 133
- “mmchconfig command” on page 170
- “mmlscluster command” on page 492
- “mmlsconfig command” on page 495
- “mmnfs command” on page 560
- “mmobj command” on page 573
- “mmsmb command” on page 709

Location

/usr/lpp/mmfs/bin

mmwatch command

Administers clustered watch folder watches.

Synopsis

```
mmwatch Device list [--events] [--watch-id WatchID] [-Y]
```

or

```
mmwatch Device list --watch-id WatchID --config [-Y]
```

or

```
mmwatch all list [--events] [-Y]
```

or

```
mmwatch all producerRestart -N { NodeName[,NodeName...] | NodeFile | NodeClass } [-q]
```

or

```
mmwatch all status [-Y]
```

or

```
mmwatch Device status [--watch-id WatchID [-v | -Y]]
```

or

```
mmwatch Device enable {-F ConfigFilePath |
          [--fileset FSetName] [--description Description]
          [--events {Event[,Event...] | ALL}]
          --event-handler handlertype
          --sink-brokers BrokerIP:Port[,BrokerIP:Port...]
          --sink-topic Topic
          [--sink-auth-config Path]}
```

or

```
mmwatch Device disable --watch-id {WatchID | all}
```

Availability

Available with IBM Spectrum Scale Advanced Edition, IBM Spectrum Scale Data Management Edition, IBM Spectrum Scale Developer Edition, or IBM Spectrum Scale Erasure Code Edition.

Description

The **mmwatch** command is used to enable, disable, list, and generally administer persistent and fault-tolerant clustered watches. The purpose of clustered watch folder is to monitor file system event activity and produce events to an external event handler, which is referenced as a sink. Errors are logged to `/var/adm/ras/mmwatch.log`, `/var/adm/ras/mmwfclient.log`, and `/var/log/messages`.

Remember: To use this command to enable, disable, and administer clustered watch folder, your cluster code level must be at IBM Spectrum Scale 5.0.3 or later, and the file system on which the clustered watch folder is set must be at IBM Spectrum Scale 5.0.2 or later.

Parameters

Device

Specifies the device name of the file system upon which the clustered watch folder configuration change or listing is to occur.

Note: You must specify the *Device* or use the **all** keyword for **mmwatch** operations.

all

Lists the active clustered watches for all file systems.

Note: You must specify the *Device* or use the **all** keyword for **mmwatch** operations.

list [--events] [-Y]

Displays the details of active clustered watches for the specified device. The optional **--events** option lists the monitored clustered watch folder events for each specific watch for the specified device. The **-Y** parameter provides output in machine-readable (colon-delimited) format.

Note: Fields that have a colon (:) are encoded to prevent confusion. For the set of characters that might be encoded, see the command documentation of **mmclidecode**. Use the **mmclidecode** command to decode the field.

producerRestart -N { *NodeName*[,*NodeName*...] | *NodeFile* | *NodeClass* }

Restarts the producers for all file systems under watch on the nodes specified by the **-N** option. The **-N** option supports a comma-separated list of nodes, a full path name to a file containing node names, or a predefined node class.

Note: Issuing this command causes the event producers for file audit logging to be restarted as well.

-q

Suppresses all **[I]** informational messages.

status

Provides the status of the nodes sending watch events to the external Kafka sink, and is subject to the following:

- If **all** is used in place of a device name, the **status** is provided for all clustered watches that are enabled within the cluster. If no clustered watches are found, a statement stating this is returned but this scenario is not considered an error.
- If a *Device* is specified without a watch ID, then the **status** is provided for all clustered watches that are enabled within the cluster associated with that device. If no clustered watches exist for the device, a statement stating this is returned and this is considered an error.
- If *Device* and **--watch-id** *WatchID* are both specified, then the **status** is provided for the single clustered watch that is associated with the given device and watch ID. If the clustered watch cannot be found, a statement stating this is returned and this is considered an error.

  Note: The **-v** flag can only be used when a watch ID is given. It provides up to the last 10 entries for the given watch ID for each component.

enable

Enables a clustered watch for the given device. The scope of the watch is the whole file system unless the **--fileset** option is provided. If **--fileset** is provided, then the watch type will be fileset watch for dependent filesets and inode space watch for independent filesets. Enablement entails setting up and validating the configuration, and applying the respective policy partition rules based on the watched events.

-F *ConfigFilePath*

Specifies the path to the configuration file.

The configuration file is populated with key:value pairs, which include all of the required parameters for **enable** and any optional fields desired in the following format:

- FILESET:<fileset name>
- EVENTS:<Event1>,<Event2> | ALL

- EVENT_HANDLER:kafkasink
- SINK_BROKERS:<sink broker:port>
- SINK_TOPIC:<sink topic>
- SINK_AUTH_CONFIG:</path/to/auth/config>
- WATCH_DESCRIPTION:<description_string>

Note: If this option is given, then the other command line options that are used to enable a watch are invalid and produce a syntax error. Using the **--fileset**, **--events**, **--event-handler**, **--sink-brokers**, **--sink-topic**, or **--sink-auth-config** options in combination with the **-F** option produces a syntax error.

--fileset ***FSetName***
Specifies the name of the fileset within the given file system to watch. If the fileset is dependent, then the watch type is *fileset*. No nested filesets are watched within a dependent fileset. If the fileset is independent, then the watch type is *inodespace*. Only nested dependent filesets are included in the watch of an independent fileset. The **--fileset** option is optional.

--description ***Description***
The watch description is an optional parameter that identifies how a watch is being used. It can be any combination of letters, numbers, and spaces, but it is limited to 50 total characters.

--config
Displays the configuration details for the specified active watch *WatchID*.

--events ***Event*[,*Event*...] | ALL}**
Defines a comma-separated string of events to monitor. The **--events** option is optional. The following events can be watched:

- IN_ACCESS
- IN_ATTRIB
- IN_CLOSE_NOWRITE
- IN_CLOSE_WRITE
- IN_CREATE
- IN_DELETE
- IN_DELETE_SELF
- IN_MODIFY
- IN_MOVED_FROM
- IN_MOVED_TO
- IN_MOVE_SELF
- IN_OPEN

If the **--events** option is not included, all events are watched.

--event-handler ***handlertype***
The only type of event handler that is currently supported is *kafkasink*. The **--event-handler** option is required.

--sink-brokers ***BrokerIP:Port*[,*BrokerIP:Port*...]**
Includes a comma-separated list of broker:port pairs for the sink Kafka cluster (the external Kafka cluster where the events are sent). The **--sink-brokers** option is required.

When specifying IP addresses in IPv6 format, you must enclose the IP address with []. For example: `[2002:90b:e006:86:19:1:186:13]:9092,[2002:90b:e006:86:19:1:186:13]:9092`.

--sink-topic ***Topic***
The topic that producers write to in the sink Kafka cluster. The **--sink-topic** option is required.

--sink-auth-config ***Path***
The full path to the file that contains the authentication details to the sink Kafka cluster. The **--sink-auth-config** option is optional. For an example of how to use this flag, see *Interaction between*

clustered watch folder and the external Kafka sink in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

disable
Disables a clustered watch for the given device. Disablement removes the configuration and deletes the policy partition rules for the watch.

Exit status

0
Successful completion.

nonzero
A failure occurs.

Security

You must have root authority to run the **mmwatch** command.

Examples

1. To list all current clustered watches that are active, issue the following command:

```
# mmwatch all list
```

A sample output is as follows:

```
# mmwatch all list
Filesystem audit1 has 1 watcher(s):
  Cluster ID            WatchID/PID     Type       Start Time                Path      Description
  ----------------------------------------------------------------------------------------------
  10019023689338312536   CLW1601406018   FSYS       Tue Sep 29 15-00-19 2020  /audit1   WatchID1234
Filesystem audit2 has no watchers.
Filesystem fs402 is at an unsupported filesystem level for watch folder.
Filesystem watch1 has 1 watcher(s):
  Cluster ID            WatchID/PID     Type       Start Time                Path      Description
  ----------------------------------------------------------------------------------------------
  10019023689338312536   CLW1601405441   FSYS       Tue Sep 29 14-50-41 2020  /watch1
```

2. To list all current clustered watches with associated events, issue the following command:

```
# mmwatch all list --events
```

A sample output is as follows:

```
# mmwatch all list --events
Filesystem audit1 has 1 watcher(s):
  Cluster ID            WatchID/PID     Monitored Events
  ------------------------------------------------------------------------------
  10019023689338312536   CLW1601409402   IN_ACCESS,IN_ATTRIB,IN_CLOSE_NOWRITE,IN_CLOSE_WRITE,
                                         IN_CREATE,IN_DELETE,IN_DELETE_SELF,IN_MODIFY,
                                         IN_MOVED_FROM,IN_MOVED_TO,IN_MOVE_SELF,
Filesystem watch1 has 1 watcher(s):
  Cluster ID            WatchID/PID     Monitored Events
  ------------------------------------------------------------------------------
  10019023689338312536   CLW1601405441   IN_ACCESS,IN_DELETE
```

3. To list the current configuration for a clustered watch, issue the following command:

```
# mmwatch Device list --watch-id WatchID --config
```

A sample output is as follows:

```
# mmwatch audit1 list --watch-id CLW1601409402 --config
WATCH_ID:CLW1601409402
DEVICE:audit1
START_TIME:Tue Sep 29 15-56-42 2020
WATCH_PATH:/audit1
EVENTS:ALL
```

```
EVENT_HANDLER:kafkasink
SINK_BROKERS:hs22n55:9092
SINK_TOPIC:SpectrumScale_154_6666561368425248142_22_FSYS_fs1_audit
SINK_AUTH_TYPE:NONE
WATCH_TYPE:FSYS
WATCH_DESCRIPTION:WatchID1234
```

4. To check the status, issue the following command:

```
# mmwatch all status
```

A sample output is as follows:

```
# mmwatch all status
Device    Watch Path                  Watch ID       Watch State
watch1    /watch1                     CLW1601405441  Active
          Node Name                                          Status
          c35f1m4n07.gpfs.net                                TIPS
          c6f2bc3n10.gpfs.net                                HEALTHY
          c6f2bc3n2.gpfs.net                                 HEALTHY
audit1    /audit1                     CLW1601409402  Active
          Node Name                                          Status
          c35f1m4n07.gpfs.net                                HEALTHY
          c6f2bc3n10.gpfs.net                                HEALTHY
          c6f2bc3n2.gpfs.net                                 HEALTHY
```

5. To enable a clustered watch using an input configuration file, issue the following command:

```
# mmwatch audit1 enable -F audit1_watch_config
```

Where `audit1_watch_config` looks like:

```
# file for creating watch on audit1
EVENTS:ALL
FILESET:ind1
EVENT_HANDLER:kafkasink
SINK_BROKERS:hs22n55.gpfs.net:9092
SINK_TOPIC:SpectrumScale_154_6666561368425248142_22_FSYS_fs1_audit
SINK_AUTH_CONFIG:audit1_auth_file
WATCH_DESCRIPTION:"This_is_my_description"
```

6. Enabling an independent fileset displays information similar to the following output:

```
# mmwatch audit1 enable --event-handler kafkasink --sink-brokers hs22n55:9092 --sink-
topic SpectrumScale_154_6666561368425248142_22_FSYS_fs1_audit --description "WatchID1234" --
fileset ind1
[I] Beginning enablement of Clustered Watch with newly created watch ID: CLW1601410684
[I] Verified the watch type is INODE for independent fileset ind1
[I] Successfully added Clustered Watch configuration file into CCR for watch: CLW1601410684
[I] Successfully added Clustered Watch configuration in the Spectrum Scale file system:
audit1 for watch: CLW1601410684
[I] Successfully added Clustered Watch policy rules for watch: CLW1601410684
[I] Successfully checked or created Clustered Watch global catchall and config fileset skip
partitions for watch: CLW1601410684
[I] Successfully checked or created Clustered Watch policy skip partitions for watch:
CLW1601410684
[I] Successfully enabled Clustered Watch: CLW1601410684
```

7. To disable a clustered watch, issue the following command:

```
# mmwatch Device disable --watch-id WatchID
```

A sample output is as follows:

```
# mmwatch audit1 disable --watch-id CLW1601410684
[I] Successfully checked or deleted Clustered Watch policy skip partitions for watch:
CLW1601410684
[I] Successfully removed Clustered Watch policy rules for watch: CLW1601410684
[I] Successfully removed Clustered Watch configuration file from the CCR for watch:
CLW1601410684
[I] Successfully removed Clustered Watch configuration directory for watch: CLW1601410684
```

```
[I] Successfully checked or removed Clustered Watch global catchall and config fileset skip
partitions for watch: CLW1601410684
[I] Successfully disabled Clustered Watch: CLW1601410684
```

See also

- “mmaudit command” on page 92

Location

/usr/lpp/mmfs/bin

mmwinservctl command

Manages the mmwinserv Windows service.

Synopsis

```
mmwinservctl set [--account AccountName [--password Password]] [--remote-shell {yes | no}]
[-N {Node[,Node...] | NodeFile | NodeClass}] [-v]
```

or

```
mmwinservctl {enable | disable | query} [-N {Node[,Node...] | NodeFile | NodeClass}] [-v]
```

Availability

Available on all IBM Spectrum Scale editions. Available on Windows.

Description

mmwinserv is a GPFS for Windows service that is needed for the proper functioning of the GPFS daemon on nodes running Windows. Optionally, the service can be configured to provide a remote execution facility for GPFS administration commands.

Use the mmwinservctl command to manage the mmwinserv service. You can set the log on account and password for the service, enable or disable the service, enable or disable the service's remote execution facility, or query its current state.

The mmwinservctl command must be run on a Windows node and it has no effect on nodes running other operating systems.

If the remote execution facility of mmwinserv is enabled, a Windows GPFS cluster can be configured to use mmwinrsh and mmwinrcp as the remote shell and remote file copy commands:

- mmwinrsh (/usr/lpp/mmfs/bin/mmwinrsh) uses Windows Named Pipes to pass the command to the target node.
- mmwinrcp (/usr/lpp/mmfs/bin/mmwinrcp) is a wrapper module that invokes the Cygwin cp command to copy the files that are needed by the mm commands. The path names on remote hosts are translated into path names based on the standard Windows ADMIN$ share.

An account must be given the right to log on as a service before it can be used to run mmwinserv. The right to log on as a service is controlled by the Local Security Policy of each Windows node. You can use the Domain Group Policy to set the Local Security Policy on all Windows nodes in a GPFS cluster.

For more information on the mmwinserv service, see *Configuring the GPFS Administration service* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Parameters

set
Sets the service configuration options and restarts the service if it is running.

enable
Sets the service to automatic startup and starts the service.

disable
Sets the service to disabled and stops the service.

query
Returns information about the service's configuration and current state.

--account *AccountName*
Specifies the log on account name for the `mmwinserv` service. By default, `mmwinserv` is configured to run using the `LocalSystem` account.

--password *Password*
Specifies the log on password for the `mmwinserv` service.

--remote-shell {yes | no}
Specifies whether or not remote connections are allowed.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}
Specifies the list of nodes on which to perform the action. The default is the node on which the `mmwinservctl` command is issued.

If the node on which the `mmwinservctl` command is issued belongs to a GPFS cluster, the nodes specified with the `-N` parameter must belong to the cluster.

If the node on which the `mmwinservctl` command is issued does not belong to a GPFS cluster, the nodes specified with the `-N` parameter must be identified by their host names or IP addresses. Node classes and node numbers cannot be used.

For general information on how to specify node names, see *Specifying nodes as input to GPFS commands* in the *IBM Spectrum Scale: Administration Guide*.

-v
Displays progress and intermediate error messages.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must be a member of the Domain Admins group to run the `mmwinservctl` command.

Examples

1. To specify 'gpfs\root' as the log on account name for the `mmwinserv` service and enable the remote command execution facility on nodes **ls21n19** and **ls21n20**, issue:

```
mmwinservctl set -N ls21n19,ls21n20 --account gpfs/root -password abcdefg –remote-shell yes
```

The system displays information similar to:

```
Node name        Service state  Remote shell  Account name
-------------------------------------------------------------
ls21n19          START_PENDING  yes           gpfs\root
ls21n20          START_PENDING  yes           gpfs\root
```

2. To display the current state of the `mmwinserv` service on all nodes in the cluster, issue:

```
mmwinservctl query -N all
```

The system displays information similar to:

```
Node name        Service state  Remote shell  Account name
-------------------------------------------------------------
ls21n19          RUNNING        yes           gpfs\root
ls21n20          RUNNING        yes           gpfs\root
ls21n14          RUNNING        yes           LocalSystem
```

Location

/usr/lpp/mmfs/bin

spectrumscale command

Installs and configures GPFS; adds nodes to a cluster; deploys and configures protocols and performance monitoring tools; configures call home and file audit logging; and upgrades GPFS and protocols.

Synopsis

```
spectrumscale setup [ -i SSHIdentity ] [ -s ServerIP ]
                    [ -st { "ss","SS","ess","ESS","ece","ECE" } ]
                    [ --storesecret ]
```

or

```
spectrumscale node add [-g] [-q] [-m] [-a] [-n] [-e] [-c] [-p] [-so] Node
```

or

```
spectrumscale node load [-g] [-q] [-m] [-a] [-n] [-e] [-c] [-p] [-so] NodeFile
```

or

```
spectrumscale node delete [-f] Node
```

or

```
spectrumscale node clear [-f]
```

or

```
spectrumscale node list
```

or

```
spectrumscale config gpfs [-l] [ -c ClusterName ] [ -p { default | randomio } ]
                          [ -r RemoteShell ] [ -rc RemoteFileCopy ]
                          [ -e EphemeralPortRange ]
```

or

```
spectrumscale config protocols [-l] [ -f FileSystem ] [ -i Interfaces ]
                               [ -m MountPoint ] [ -e ExportIPPool ]
```

or

```
spectrumscale config object [ -f FileSystem ] [ -m MountPoint ][ -e EndPoint ]
                            [ -o ObjectBase ] [ -i InodeAllocation ]
                            [ -au AdminUser ] [ -ap AdminPassword ]
                            [ -su SwiftUser ] [ -sp SwiftPassword ]
                            [ -dp DatabasePassword ]
                            [ -s3 { on | off } ]
```

Important: The Object protocol requires OpenStack 16 repositories to be available to satisfy the necessary dependencies. If the systems have the *Red Hat OpenStack Platform* subscription, you can add the repositories by using the following commands:

```
subscription-manager attach --pool=PoolID
subscription-manager repos --enable=openstack-16-for-rhel-8-Architecture-rpms
subscription-manager repos --enable=codeready-builder-for-rhel-8-Architecture-rpms
```

You can obtain the subscription *PoolID* by listing the product subscription information. Replace *Architecture* with x86_64 or ppc64le as appropriate.

The OpenStack packages are also available from publicly accessible repositories. For instructions on setting up alternate repositories for the OpenStack 16 (Train) packages, see OpenStack packages for RHEL and CentOS.

or

```
spectrumscale config hdfs new -n Name -nn NameNodes -dn DataNodes
                              -f FileSystem -d DataDir
```

or

```
spectrumscale config hdfs import -l LocalDir
```

or

```
spectrumscale config hdfs add -n Name [ -nn NameNodes ] [ -dn DataNodes ]
```

or

```
spectrumscale config hdfs list
```

or

```
spectrumscale config hdfs clear [-f]
```

or

```
spectrumscale config perfmon [ -r { on | off } ] [ -d { on | off } ] [-l]
```

or

```
spectrumscale config clear { gpfs | protocols | object }
```

or

```
spectrumscale config update
```

or

```
spectrumscale config populate --node Node
```

or

```
spectrumscale nsd add -p Primary [ -s Secondary ] [ -fs FileSystem ] [ -po Pool ]
                      [ -u { dataOnly | dataAndMetadata | metadataOnly | descOnly | localCache } ]
                      [ -fg FailureGroup ] [ --no-check ]
                      PrimaryDevice [ PrimaryDevice ... ]
```

or

```
spectrumscale nsd delete NSD
```

or

```
spectrumscale nsd modify [ -n Name ]
                         [ -u { dataOnly | dataAndMetadata | metadataOnly | descOnly | localCache } ]
                         [ -po Pool ] [ -fs FileSystem ] [ -fg FailureGroup ]
                         NSD
```

or

```
spectrumscale nsd servers { add | delete | setprimary } -s NSDServer NSDs
```

or

```
spectrumscale nsd clear [-f]
```

or

```
spectrumscale nsd list
```

or

```
spectrumscale filesystem modify [ -B { 64K | 128K | 256K | 512K | 1M | 2M | 4M | 8M | 16M } ]
                                [ -m MountPoint]
                                [ -r { 1 | 2 | 3 } ] [ -mr { 1 | 2 | 3 } ]
                                [ -MR { 1 | 2 | 3} ] [ -R { 1 | 2 | 3 } ]
                                [ --metadata_block_size {64K | 128K | 256K | 512K | 1M | 2M | 4M | 8M | 16M}]
                                [ --fileauditloggingenable
                                [ --degradedperformance] [ --degradedperformancedisable ] ]
                                [ --fileauditloggingdisable] [--logfileset LogFileset]
                                [ --retention RetentionPeriod] FileSystem
```

or

```
spectrumscale filesystem define [-fs FileSystem] -vs VdiskSet [--mmcrfs MmcrfsParams]
```

or

```
spectrumscale filesystem list
```

or

```
spectrumscale fileauditlogging enable
```

or

```
spectrumscale fileauditlogging disable
```

or

```
spectrumscale fileauditlogging list
```

or

```
spectrumscale recoverygroup define [ -rg RGName ] [ -nc ScaleOutNodeClassName ] --node Node
```

or

```
spectrumscale recoverygroup undefine RGName
```

or

```
spectrumscale recoverygroup change [ -rg NewRGName ] ExistingRGName
```

or

```
spectrumscale recoverygroup list
```

or

```
spectrumscale recoverygroup clear [-f]
```

or

```
spectrumscale vdiskset define [ -vs VdiskSet ] [ -rg RGName ]
                              -code { 4+2P | 4+3P | 8+2P | 8+3P }
```

```
                                -bs { 1M | 2M | 4M | 8M | 16M }
                                -ss VdiskSetSize
```

or

```
spectrumscale vdiskset undefine VdiskSet
```

or

```
spectrumscale vdiskset  clear [-f]
```

or

```
spectrumscale vdiskset list
```

or

```
spectrumscale callhome enable
```

or

```
spectrumscale callhome disable
```

or

```
spectrumscale callhome config -n CustName -i CustID -e CustEmail -cn CustCountry
                                [ -s ProxyServerIP ] [ -pt ProxyServerPort ]
                                [ -u ProxyServerUserName ] [ -pw ProxyServerPassword ] [-a]
```

or

```
spectrumscale callhome clear  { --all | -n | -i | -e | -cn | -s | -pt | -u | -pw }
```

or

```
spectrumscale callhome schedule  { -d | -w } [-c]
```

or

```
spectrumscale callhome list
```

or

```
spectrumscale enable { object | nfs | smb | hdfs }
```

or

```
spectrumscale disable { object | nfs | smb | hdfs }
```

CAUTION: Disabling object service discards OpenStack Swift configuration and ring files from the CES cluster. If OpenStack Keystone configuration is configured locally, disabling object storage also discards the Keystone configuration and database files from the CES cluster. However, the data is not removed. For subsequent object service enablement with a clean configuration and new data, remove object store fileset and set up object environment. See the mmobj swift base command. For more information, contact the IBM Support Center.

or

```
spectrumscale install [-pr] [-po] [-s SecretKey] [-f] [--skip]
```

or

```
spectrumscale deploy [-pr] [-po] [-s SecretKey] [-f] [--skip]
```

or

```
spectrumscale upgrade precheck [ --skip ]
```

or

```
spectrumscale upgrade config offline [ -N Node ] [ --clear ]
                             exclude [ -N Node ] [ --clear ]
                             list
                             clear
```

or

```
spectrumscale upgrade config workload [ -l | -p { on | off } ]
```

or

```
spectrumscale upgrade run [ --skip ]
```

or

```
spectrumscale upgrade postcheck [ --skip ]
```

or

```
spectrumscale upgrade showversions
```

Availability

Available on all IBM Spectrum Scale editions.

Description

Use the `spectrumscale` command (the installation toolkit) to do the following:

- Install and configure GPFS.
- Add GPFS nodes to an existing cluster.
- Deploy and configure SMB, NFS, object (OpenStack Swift), HDFS, and performance monitoring tools on top of GPFS.
- Enable and configure the file audit logging function.
- Enable and configure the call home function.
- Configure recovery groups and vdisk sets, and define file systems for an IBM Spectrum Scale Erasure Code Edition environment.
- Upgrade IBM Spectrum Scale components.
  - Perform offline upgrade of nodes that have services that are down or stopped.
  - Exclude one or more nodes from the current upgrade run.
  - Resume an upgrade run after a failure.

Note:

The following prerequisites and assumptions apply:

- The installation toolkit requires following packages:
  - Python 3.6
  - net-tools

  - elfutils-libelf-devel [Only on Red Hat Enterprise Linux 8.x nodes with kernel version 4.15 or later]
  - Ansible® 2.9.15
- TCP traffic from the nodes should be allowed through the firewall to communicate with the installation toolkit on port 10080 for package distribution.
- The nodes themselves have external Internet access or local repository replicas that can be reached by the nodes to install necessary packages (dependency installation). For more information, see the *Repository setup* section of the *Installation prerequisites* topic in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.
- To install protocols, there must a GPFS cluster running a minimum version of 4.1.1.0 with CCR enabled.
- The node that you plan to run the installation toolkit from must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages.
- Any node that is set up to be a call home node must have network connectivity to IBM Support to upload data.

The installation toolkit performs verification during the precheck phase to ensure that passwordless SSH is set up correctly. This verification includes:

- Check whether passwordless SSH is set up between all admin nodes and all the other nodes in the cluster. If this check fails, a fatal error occurs.
- Check whether passwordless SSH is set up between all protocol nodes and all the other nodes in the cluster. If this check fails, a warning is displayed.
- Check whether passwordless SSH is set up between all protocol nodes in the cluster. If this check fails, a fatal error occurs.

Parameters

setup
Configures the installer node in the cluster definition file. The IP address passed must be of the node from which the installation toolkit will be run. The SSH key passed must be the key that the installation toolkit uses to have passwordless SSH onto all other nodes. This is the first command you run to set up IBM Spectrum Scale. This option accepts the following arguments:

-i *SSHIdentity*
Adds the path to the SSH identity file into the configuration.

-s *ServerIP*
Adds the control node IP into the configuration.

-st {"ss","SS","ess","ESS","ece","ECE"}
Specifies the setup type.

The allowed values are `ess`, `ece`, and `ss`. The default value is `ss`.

- If you are using the installation toolkit in a cluster containing ESS, specify the setup type as `ess`.
- If you are using the installation toolkit in an IBM Spectrum Scale Erasure Code Edition cluster, specify the setup type as `ece`.
- The setup type `ss` specifies an IBM Spectrum Scale cluster containing no ESS nodes.

Regardless of the mode, the installation toolkit contains safeguards to prevent changing of a tuned ESS configuration. While adding a node to the installation toolkit, it looks at whether the node is currently in an existing cluster and if so, it checks the node class. ESS I/O server nodes are detected based upon existence within the `gss_ppc64` node class. ESS EMS nodes are detected based upon existence within the `ems` node class. ESS I/O server nodes are not allowed to be added to the installation toolkit and must be managed by the ESS toolsets contained in the EMS node. A single ESS EMS node is allowed to be added to the installation toolkit. Doing so adds this node as an admin node of the installation toolkit functions. While the installation toolkit runs from a non-ESS node, it uses the designated admin node (an EMS node in this case) to run mm

commands on the cluster as a whole. Once in the ESS mode, the following assumptions and restrictions apply:

- File audit logging is not configurable using the installation toolkit.
- Call home is not configurable using the installation toolkit
- EMS node will be the only admin node designated in the installation toolkit. This designation will automatically occur when the EMS node is added.
- EMS node will be the only GUI node allowed in the installation toolkit. Additional existing GUI nodes can exist but they cannot be added.
- EMS node will be the only performance monitoring collector node allowed within the installation toolkit. Additional existing collectors can exist but they cannot be added.
- EMS node cannot be designated as an NSD or a protocol node.
- I/O server nodes cannot be added to the installation toolkit. These nodes must be managed outside the installation toolkit by ESS toolsets contained in the EMS node.
- NSDs and file systems managed by the I/O server nodes cannot be added to the installation toolkit.
- File systems managed by the I/O server nodes can be used for placement of the Object fileset as well as CESSharedRoot file system. Simply point the installation toolkit to the path.
- The cluster name is set upon addition of the EMS node to the installation toolkit. It is determined by **mmlscluster** being run from the EMS node.
- EMS node must have passwordless SSH set up to all nodes, including any protocol, NSD, and client nodes being managed by the installation toolkit.
- EMS node can be a different architecture or operating system than the protocol, NSD, and client nodes being managed by the installation toolkit.
- If the config populate function is used, an EMS node of a different architecture or operating system than the protocol, NSD, and client nodes can be used.
- If the config populate function is used, a mix of architectures within the non-ESS nodes being added or currently within the cluster cannot be used. To handle this case, use the installation toolkit separately for each architecture grouping. Run the installation toolkit from a node with similar architecture to add the required nodes. Add the EMS node and use the setup type `ess`.

--storesecret
Disables the prompts for the encryption secret. Generates a random encryption secret and stores it in the cluster definition file.

CAUTION: If you use this option, passwords will not be securely stored.

You can override the encryption secret stored in the cluster definition file by using the **-s** *SecretKey* option with the **./spectrumscale install** or the **./spectrumscale deploy** command.

node
Used to add, remove, or list nodes in the cluster definition file. This command only interacts with this configuration file and does not directly configure nodes in the cluster itself. The nodes that have an entry in the cluster definition file will be used during install, deploy, or upgrade. This option accepts the following arguments:

add *Node*
Adds the specified node and configures it according to the following arguments:

-g
Adds GPFS Graphical User Interface servers to the cluster definition file.

-q
Configures the node as a quorum node.

-m
Configures the node as a manager node.

-a
Configures the node as an admin node.

-n
Specifies the node as NSD.

-e
Specifies the node as the EMS node of an ESS system. This node is automatically specified as the admin node.

-c
Specifies the node as a call home node.

-p
Configures the node as a protocol node.

-so
Specifies the node as a scale-out node. The setup type must be ece for adding this type of nodes in the cluster definition.

Node
Specifies the node name.

load *NodeFile*
Loads the specified file containing a list of nodes, separated per line; adds the nodes specified in the file and configures them according to the following:

-g
Sets the nodes as GPFS Graphical User Interface server.

-q
Sets the nodes as quorum nodes.

-m
Sets the nodes as manager nodes.

-a
Sets the nodes as admin nodes.

-n
Sets the nodes as NSD servers.

-e
Sets the node as the EMS node of an ESS system. This node is automatically specified as the admin node.

-c
Sets the nodes as call home nodes.

-p
Sets the nodes as protocol nodes.

-so
Sets the nodes as scale-out nodes. The setup type must be ece for adding this type of nodes in the cluster definition.

delete *Node*
Removes the specified node from the configuration. The following option is accepted.

-f
Forces the action without manual confirmation.

clear
Clears the current node configuration. The following option is accepted:

-f
Forces the action without manual confirmation.

list
Lists the nodes configured in your environment.

config
Used to set properties in the cluster definition file that will be used during install, deploy, or upgrade. This command only interacts with this configuration file and does not directly configure these properties on the GPFS cluster. This option accepts the following arguments:

gpfs
Sets any of the following GPFS-specific properties to be used during GPFS installation and configuration:

-l
Lists the current settings in the configuration.

-c *ClusterName*

-p
Specifies the profile to be set on cluster creation. The following values are accepted:

default
Specifies that the `GpfsProtocolDefaults` profile is to be used.

randomio
Specifies that the `GpfsProtocolRandomIO` profile is to be used.

-r *RemoteShell*
Specifies the remote shell binary to be used by GPFS. If no remote shell is specified in the cluster definition file, `/usr/bin/ssh` will be used as the default.

-rc *RemoteFileCopy*
Specifies the remote file copy binary to be used by GPFS. If no remote file copy binary is specified in the cluster definition file, `/usr/bin/scp` will be used as the default.

-e *EphemeralPortRange*

Specifies an ephemeral port range to be set on all GPFS nodes. If no port range is specified in the cluster definition, 60000-61000 is used as default.

For information about ephemeral port range, see the topic about GPFS port usage in the *Miscellaneous advanced administration tasks* in *IBM Spectrum Scale: Administration Guide*.

protocols
Provides details of the GPFS environment that will be used during protocol deployment, according to the following options:

-l
Lists the current settings in the configuration.

-f *FileSystem*
Specifies the file system.

-m *MountPoint*
Specifies the shared file system mount point or path.

-e *ExportIPPool*
Specifies a comma-separated list of additional CES export IP addresses to configure on the cluster.

In the CES interface mode, you must specify the CES IP addresses with the installation toolkit in the Classless Inter-Domain Routing (CIDR) notation. In the CIDR notation, the IP address is followed by a forward slash and the prefix length.

```
IPAddress/PrefixLength
```

For example,

IPv6
`2001:0DB8::/32`

IPv4
192.0.2.0/20

You must specify the prefix length for every CES IP address, otherwise you cannot add the IP address by using the installation toolkit.

- When you are using IPv6 addresses, prefix length must be in the range 1 - 124.
- When you are using IPv4 addresses, prefix length must be in the range 1 - 30.

-i *Interfaces*
Specifies a comma-separated list of network interfaces.

object
Sets any of the following Object-specific properties to be used during Object deployment and configuration:

-l
Lists the current settings in the configuration.

-f *FileSystem*
Specifies the file system.

-m *MountPoint*
Specifies the mount point.

-e *EndPoint*
Specifies the host name that will be used for access to the object store. This should be a round-robin DNS entry that maps to all CES IP addresses or the address of a load balancer front end; this will distribute the load of all keystone and object traffic that is routed to this host name. Therefore the endpoint is an IP address in a DNS or in a load balancer that maps to a group of export IPs (that is, CES IPs that were assigned on the protocol nodes).

-o *ObjectBase*
Specifies the object base.

-i *InodeAllocation*
Specifies the inode allocation.

-au *AdminUser*
Specifies the user name for the admin.

-ap *AdminPassword*
Specifies the admin user password. This credential is for the Keystone administrator. This user can be local or on remote authentication server based on the authentication type used.

Note: You will be prompted to enter a Secret Encryption Key which will be used to securely store the password. Choose a memorable pass phrase which you will be prompted for each time you enter the password.

-su *SwiftUser*
Specifies the Swift user name. This credential is for the Swift services administrator. All Swift services are run in this user's context. This user can be local or on remote authentication server based on the authentication type used.

-sp *SwiftPassword*
Specifies the Swift user password.

Note: You will be prompted to enter a Secret Encryption Key which will be used to securely store the password. Choose a memorable pass phrase which you will be prompted for each time you enter the password.

-dp *DataBasePassword*
Specifies the object database user password.

Note: You will be prompted to enter a Secret Encryption Key which will be used to securely store the password. Choose a memorable pass phrase which you will be prompted for each time you enter the password.

-s3 on | off
Specifies whether s3 is to be turned on or off.

hdfs
Sets Hadoop Distributed File System (HDFS) related configuration in the cluster definition file:

new
Defines configuration for a new HDFS cluster.

-n *Name*
Specifies the name of the new HDFS cluster.

Note: The name of the HDFS cluster must not contain any upper case letters.

-nn *NameNodes*
Specifies the name node host names in a comma-separated list.

-dn *DataNodes*
Specifies the data node host names in a comma-separated list.

-f *FileSystem*
Specifies the IBM Spectrum Scale file system name.

-d *DataDir*
Specifies the IBM Spectrum Scale data directory name.

Note: You must specify only the directory name. Do not specify the directory path.

import
Imports an existing HDFS configuration.

-l *LocalDir*
Specifies the local directory that contains the existing HDFS configuration.

add
Adds name nodes or data nodes in an existing HDFS configuration.

-n *Name*
Specifies the name of the existing HDFS cluster.

-nn *NameNodes*
Specifies the name node host names in a comma-separated list.

-dn *DataNodes*
Specifies the data node host names in a comma-separated list.

list
Lists the current HDFS configuration.

clear
Clears the current HDFS configuration from the cluster definition file.

-f
Forces action without manual confirmation.

perfmon
Sets performance monitoring specific properties to be used during installation and configuration:

-r on | off

Specifies if the install toolkit can reconfigure performance monitoring.

Note: When set to on, reconfiguration might move the collector to different nodes and it might reset sensor data. Custom sensors and data might be erased.

-d on | off

Specifies if performance monitoring should be disabled (not installed).

Note: When set to on, pmcollector and pmsensor packages are not installed or upgraded. Existing sensor or collector state remains as is.

-l
Lists the current settings in the configuration.

clear
Removes specified properties from the cluster definition file:

gpfs
Removes GPFS related properties from the cluster definition file:

-c
Clears the GPFS cluster name.

-p
Clears the GPFS profile set in the cluster definition file and sets it to `default`.

-r *RemoteShell*
Clears the absolute path name of the remote shell command GPFS uses for node communication. For example, `/usr/bin/ssh`.

-rc *RemoteFileCopy*
Clears the absolute path name of the remote copy command GPFS uses when transferring files between nodes. For example, `/usr/bin/scp`.

-e *EphemeralPortRange*
Clears the GPFS daemon communication port range.

--all
Clears all settings in the cluster definition file.

protocols
Removes protocols related properties from the cluster definition file:

-f
Clears the shared file system name.

-m
Clears the shared file system shared file system mount point or path.

-e
Clears a comma-separated list of additional CES export IPs to configure on the cluster.

--all
Clears all settings in the cluster definition file.

object
Removes object related properties from the cluster definition file:

-f
Clears the object file system name.

-m
Clears the absolute path to your file system on which the objects reside.

-e
Clears the host name which maps to all CES IP addresses in a round-robin manner.

-o
Clears the GPFS fileset to be created or used as the object base.

-i
Clears the GPFS fileset inode allocation to be used by the object base.

-t
Clears the admin token to be used by Keystone.

-au
Clears the user name for the admin user.

-ap
Clears the password for the admin user.

-su
Clears the user name for the Swift user.

-sp
Clears the password for the Swift user.

-dp
Clears the password for the object database.

-s3
Clears the S3 API setting, if it is enabled.

--all

Clears all settings in the cluster definition file.

update
Updates operating system and CPU architecture fields in the cluster definition file. This update is automatically done if you run the upgrade precheck while upgrading to IBM Spectrum Scale release 4.2.2 or later.

populate
Populates the cluster definition file with the current cluster state. In the following upgrade scenarios, you might need to update the cluster definition file with the current cluster state:

- A manually created cluster in which you want to use the installation toolkit to perform administration tasks on the cluster such as adding protocols, adding nodes, and upgrading.
- A cluster created using the installation toolkit in which manual changes were done without using the toolkit wherein you want to synchronize the installation toolkit with the updated cluster configuration that was performed manually.

--node *Node*
Specifies an existing node in the cluster that is used to query the cluster information. If you want to use the **spectrumscale config populate** command to retrieve data from a cluster containing ESS, you must specify the EMS node with the --node flag.

nsd
Used to add, remove, or list NSDs, as well as add file systems in the cluster definition file. This command only interacts with this configuration file and does not directly configure NSDs on the cluster itself. The NSDs that have an entry in the cluster definition file will be used during install. This option accepts the following arguments:

add
Adds an NSD to the configuration, according to the following specifications:

-p *Primary*
Specifies the primary NSD server name.

-s *Secondary*
Specifies the secondary NSD server names. You can use a comma-separated list to specify up to seven secondary NSD servers.

-fs *FileSystem*
Specifies the file system to which the NSD is assigned.

-po *Pool*
Specifies the file system pool.

-u
Specifies NSD usage. The following values are accepted:

dataOnly

dataAndMetadata

metadataOnly

descOnly

localCache

-fg *FailureGroup*
Specifies the failure group to which the NSD belongs.

--no-check
Specifies not to check for the device on the server.

PrimaryDevice
Specifies the device name on the primary NSD server.

delete *NSD*
Removes the specified NSD from the configuration.

modify *NSD*
Modifies the NSD parameters on the specified NSD, according to the following options:

-n *Name*
Specifies the name.

-u
The following values are accepted:

dataOnly

dataAndMetadata

metadataOnly

descOnly

localCache

-po *Pool*
Specifies the pool

-fs *FileSystem*
Specifies the file system.

-fg *FailureGroup*
Specifies the failure group.

servers
Adds and removes servers, and sets the primary server for NSDs.

add -s *NSDServer NSDs*
Adds an NSD server for the specified NSDs.

remove -s *NSDServer NSDs*
Removes an NSD server from the server list for the specified NSDs.

setprimary -s *NSDServer NSDs*
Sets a new primary server for the specified NSDs.

clear
Clears the current NSD configuration. The following option is accepted:

-f
Forces the action without manual confirmation.

list
Lists the NSDs configured in your environment.

filesystem
Used to list or modify file systems in the cluster definition file. This command only interacts with this configuration file and does not directly modify file systems on the cluster itself. To modify the properties of a file system in the cluster definition file, the file system must first be added with `spectrumscale nsd`. This option accepts the following arguments:

modify
Modifies the file system attributes. This option accepts the following arguments:

-B
Specifies the file system block size. This argument accepts the following values: 64K, 128K, 256K, 512K, 1M, 2M, 4M, 8M, 16M.

-m *MountPoint*
Specifies the mount point.

-r
Specifies the number of copies of each data block for a file. This argument accepts the following values: 1, 2, 3.

-mr
Specifies the number of copies of inodes and directories. This argument accepts the following values: 1, 2, 3.

-MR
Specifies the default maximum number of copies of inodes and directories. This argument accepts the following values: 1, 2, 3.

-R
Specifies the default maximum number of copies of each data block for a file. This argument accepts the following values: 1, 2, 3.

--metadata_block_size
Specifies the file system meta data block size. This argument accepts the following values: 64K, 128K, 256K, 512K, 1M, 2M, 4M, 8M, 16M.

--fileauditloggingenable
Enables file audit logging on the specified file system.

--fileauditloggingdisable
Disables file audit logging on the specified file system.

--logfileset *LogFileset*
Specifies the log fileset name for file audit logging. The default value is `.audit_log`.

--retention *RetentionPeriod*
Specifies the file audit logging retention period in number of days. The default value is 365 days.

FileSystem
Specifies the file system to be modified.

define
Adds file system attributes in an IBM Spectrum Scale Erasure Code Edition environment. The setup type must be `ece` for using this option.

Note: If you are planning to deploy protocols in the IBM Spectrum Scale Erasure Code Edition cluster, you must define a CES shared root file system before initiating the installation toolkit deployment phase by using the following command.

```
./spectrumscale config protocols -f FileSystem -m MountPoint
```

-fs *FileSystem*
Specifies the file system to which the vdisk set is to be assigned.

-vs *VdiskSet*
Specifies the vdisk sets to be affected by a file system operation.

--mmcrfs *MmcrfsParams*
Specifies that all command line parameters following the **`--mmcrfs`** flag must be passed to the IBM Spectrum Scale **`mmcrfs`** command and they must not be interpreted by the **`mmvdisk`** command.

list
Lists the file systems configured in your environment.

fileauditlogging
Enable, disable, or list the file audit logging configuration in the cluster definition file.

enable
Enables the file audit logging configuration in the cluster definition file.

disable
Disables the file audit logging configuration in the cluster definition file.

list
Lists the file audit logging configuration in the cluster definition file.

recoverygroup
Define, undefine, change, list, or clear recovery group related configuration in the cluster definition file in an IBM Spectrum Scale Erasure Code Edition environment. The setup type must be ece for using this option.

define
Defines recovery groups in the cluster definition file.

-rg *RgName*
Sets the name of the recovery group.

-nc *ScaleOutNodeClassName*
Sets the name of the scale-out node class.

--node *Node*
Specifies the scale-out node with in an existing IBM Spectrum Scale Erasure Code Edition cluster for the server node class.

undefine
Undefines specified recovery group from the cluster definition file.

RgName
The name of the recovery group that is to be undefined.

change
Changes the recovery group name.

ExistingRgName
The name of the recovery group that is to be modified.

-rg *NewRgName*
The new name of the recovery group.

clear
Clears the current recovery group configuration from the cluster definition file.

-f
Forces operation without manual confirmation.

list
Lists the current recovery group configuration in the cluster definition file.

vdiskset
Define, undefine, list, or clear vdisk set related configuration in the cluster definition file in an IBM Spectrum Scale Erasure Code Edition environment. The setup type must be ece for using this option.

define
Defines vdisk sets in the cluster definition file.

-vs *VdiskSet*
Sets the name of the vdisk set.

-rg *RgName*
Specifies an existing recovery group with which the defined vdisk set is to be associated.

-code
Defines the erasure code. This argument accepts the following values: 4+2P, 4+3P, 8+2P, and 8+3P.

-bs
Specifies the block size for a vdisk set definition. This argument accepts the following values: 1M, 2M, 4M, 8M, and 16M.

-ss *VdiskSetSize*
Defines the vdisk set size in percentage of the available storage space.

undefine
Undefines specified vdisk set from the cluster definition file.

VdiskSet
The name of the vdisk set that is to be undefined.

clear
Clears the current vdisk set configuration from the cluster definition file.

-f
Forces operation without manual confirmation.

list
Lists the current vdisk set configuration in the cluster definition file.

callhome
Used to enable, disable, configure, schedule, or list call home configuration in the cluster definition file.

enable
Enables call home in the cluster definition file.

disable
Disables call home in the cluster definition file. The call home function is enabled by default in the cluster definition file. If you disable it in the cluster definition file, the call home packages are installed on the nodes but no configuration is done by the installation toolkit.

config
Configures call home settings in the cluster definition file.

-n *CustName*
Specifies the customer name for the call home configuration.

-i *CustID*
Specifies the customer ID for the call home configuration.

-e *CustEmail*
Specifies the customer email address for the call home configuration.

-cn *CustCountry*
Specifies the customer country code for the call home configuration.

-s *ProxyServerIP*
Specifies the proxy server IP address for the call home configuration. This is an optional parameter.

If you are specifying the proxy server IP address, the proxy server port must also be specified.

-pt *ProxyServerPort*
Specifies the proxy server port for the call home configuration. This is an optional parameter.

If you are specifying the proxy server port, the proxy server IP address must also be specified.

-u *ProxyServerUserName*
Specifies the proxy server user name for the call home configuration. This is an optional parameter.

-pw *ProxyServerPassword*
Specifies the proxy server password for the call home configuration. This is an optional parameter.

If you do not specify a password on the command line, you are prompted for a password.

-a
When you specify the call home configuration settings by using the **./spectrumscale callhome config**, you are prompted to accept or decline the support information collection message. Use the -a parameter to accept that message in advance. This is an optional parameter.

If you do not specify the -a parameter on the command line, you are prompted to accept or decline the support information collection message.

Clear
Clears the specified call home settings from the cluster definition file.

--all
Clears all call home settings from the cluster definition file.

-n
Clears the customer name from the call home configuration in the cluster definition file.

-i
Clears the customer ID from the call home configuration in the cluster definition file.

-e
Clears the customer email address from the call home configuration in the cluster definition file.

-cn
Clears the customer country code from the call home configuration in the cluster definition file.

-s
Clears the proxy server IP address from the call home configuration in the cluster definition file.

-pt
Clears the proxy server port from the call home configuration in the cluster definition file.

-u
Clears the proxy server user name from the call home configuration in the cluster definition file.

-pw
Clears the proxy server password from the call home configuration in the cluster definition file.

schedule
Specifies the call home data collection schedule in the cluster definition file.

By default, the call home data collection is enabled in the cluster definition file and it is set for a daily and a weekly schedule. Daily data uploads are by default executed at 02:xx AM each day. Weekly data uploads are by default executed at 03:xx AM each Sunday. In both cases, xx is a random number from 00 to 59. You can use the **spectrumscale callhome schedule** command to set either a daily or a weekly call home data collection schedule.

-d
Specifies a daily call home data collection schedule.

If call home data collection is scheduled daily, data uploads are executed at 02:xx AM each day. xx is a random number from 00 to 59.

-w
Specifies a weekly call home data collection schedule.

If call home data collection is scheduled weekly, data uploads are executed at 03:xx AM each Sunday. xx is a random number from 00 to 59.

-c
Clears the call home data collection schedule in the cluster definition file.

The call home configuration can still be applied without a schedule being set. In that case, you either need to manually run and upload data collections or you can set the call home

schedule to the desired interval at a later time with Daily: **./spectrumscale callhome schedule -d**, Weekly: **./spectrumscale callhome schedule -w**, or Both Daily and Weekly: **./spectrumscale callhome schedule -d -w** commands.

list
Lists the call home configuration specified in the cluster definition file.

enable
Used to enable Object, SMB, HDFS, or NFS in the cluster definition file. This command only interacts with this configuration file and does not directly enable any protocols on the GPFS cluster itself. The default configuration is that all protocols are disabled. If a protocol is enabled in the cluster definition file, this protocol will be enabled on the GPFS cluster during deploy. This option accepts the following arguments:

obj
Object

nfs
NFS

hdfs
HDFS

smb
SMB

disable
Used to disable Object, SMB, HDFS, or NFS in the cluster definition file. This command only interacts with this configuration file and does not directly disable any protocols on the GPFS cluster itself. The default configuration is that all protocols are disabled, so this command is only necessary if a protocol has previously been enabled in the cluster definition file, but is no longer required.

Note: Disabling a protocol in the cluster definition will not disable this protocol on the GPFS cluster during a deploy, it merely means that this protocol will not be enabled during a deploy.

This option accepts the following arguments:

obj
Object

CAUTION: Disabling object service discards OpenStack Swift configuration and ring files from the CES cluster. If OpenStack Keystone configuration is configured locally, disabling object storage also discards the Keystone configuration and database files from the CES cluster. However, the data is not removed. For subsequent object service enablement with a clean configuration and new data, remove object store fileset and set up object environment. See the mmobj swift base command. For more information, contact the IBM Support Center.

nfs
NFS

hdfs
HDFS

smb
SMB

install
Installs, creates a GPFS cluster, creates NSDs and adds nodes to an existing GPFS cluster. The installation toolkit will use the environment details in the cluster definition file to perform these tasks. If all configuration steps have been completed, this option can be run with no arguments (and pre-install and post-install checks will be performed automatically).

For a "dry-run," the following arguments are accepted:

-pr
Performs a pre-install environment check.

-po
Performs a post-install environment check.

-s *SecretKey*
Specifies the secret key on the command line required to decrypt sensitive data in the cluster definition file and suppresses the prompt for the secret key.

If you do not specify a secret key, the installation toolkit uses the encryption secret stored in the cluster definition file for decryption. By using this option, you can override the encryption secret stored in the cluster definition file that is generated by using the **--storesecret** option with the **./spectrumscale setup** command.

-f
Forces action without manual confirmation.

--skip
Bypasses the specified precheck and suppresses prompts. For example, specifying **--skip ssh** bypasses the SSH connectivity check.

deploy
Deploys protocols on an existing GPFS cluster. The installation toolkit uses the environment details in the cluster definition file to perform these tasks. If all configuration steps are completed, this option can be run with no arguments (and pre-deploy and post-deploy checks are performed automatically). However, the secret key is prompted for unless it is passed as an argument by using the **-s** flag.

For a "dry-run," the following arguments are accepted:

-pr
Performs a pre-deploy environment check.

-po
Performs a post-deploy environment check.

-s *SecretKey*
Specifies the secret key on the command line required to decrypt sensitive data in the cluster definition file and suppresses the prompt for the secret key.

If you do not specify a secret key, the installation toolkit uses the encryption secret stored in the cluster definition file for decryption. By using this option, you can override the encryption secret stored in the cluster definition file that is generated by using the **--storesecret** option with the **./spectrumscale setup** command.

-f
Forces action without manual confirmation.

--skip
Bypasses the specified precheck and suppresses prompts. For example, specifying **--skip ssh** bypasses the SSH connectivity check.

upgrade
Performs upgrade procedure, upgrade precheck, upgrade postcheck, and upgrade related configuration to add nodes as offline, or exclude nodes from the upgrade run.

precheck
Performs health checks on the cluster prior to the upgrade.

During the upgrade precheck, the installation toolkit displays messages in a number of scenarios including:

- If there are AFM relationships in the cluster. All file systems that have associated AFM primary or cache filesets are listed and reference to procedure for stopping and restarting replication is provided.

config
Manage upgrade related configuration in the cluster definition file.

offline
Designates specified nodes in the cluster as offline for the upgrade run. For entities designated as offline, only the packages are upgraded during the upgrade; the services are not restarted after the upgrade. You can use this option to designate those nodes as offline that have services down or stopped, or that have unhealthy components that are flagged in the upgrade precheck.

-N *Node*
You can specify one or more nodes that are a part of the cluster that is being upgraded with -N in a comma-separated list. For example: `node1,node2,node3`

If the nodes being specified as offline are protocol nodes then all components (GPFS, SMB, NFS, and object) are added as offline in the cluster configuration. If the nodes being specified as offline are not protocol nodes then GPFS is added as offline in the cluster configuration.

--clear
Clears the offline nodes information from the cluster configuration.

exclude
Designates specified nodes in a cluster to be excluded from the upgrade run. For nodes designated as excluded, the installation toolkit does not perform any action during the upgrade. This option allows you to upgrade a subset of a cluster.

Note: Nodes that are designated as excluded must be upgraded at a later time to complete the cluster upgrade.

-N *Node*
You can specify one or more nodes that are a part of the cluster that is being upgraded with -N in a comma-separated list. For example: `node1,node2,node3`

--clear
Clears the excluded nodes information from the cluster configuration.

workload
Sets the installation toolkit to prompt users to shut down their workloads before an upgrade on each node defined in the cluster definition file.

-p { on | off }
Enables or disables the prompt to users to shut down their workloads before an upgrade.

-l
Lists the current workload prompt related configuration in the cluster definition file.

list
Lists the upgrade related configuration information in the cluster definition file.

clear
Clears the upgrade related configuration in the cluster definition file.

run
Upgrades components of an existing IBM Spectrum Scale cluster.

This command can still be used even if all protocols are not enabled. If a protocol is not enabled, then the respective packages are still upgraded, but the respective service is not started. The installation toolkit uses environment details in the cluster definition file to perform upgrade tasks.

The installation toolkit includes the ability to determine if an upgrade is being run for the first time or if it is a rerun of a failed upgrade.

To perform environment health checks prior to and after the upgrade, run the **`./spectrumscale upgrade`** command using the `precheck` and `postcheck` arguments. This is not required, however, because specifying upgrade run with no arguments also runs these checks.

--skip
Bypasses the specified precheck and suppresses prompts. For example, specifying **`--skip ssh`** bypasses the SSH connectivity check.

postcheck
Performs health checks on the cluster after the upgrade has been completed.

showversions
Shows installed versions of GPFS and protocols and available versions of these components in the configured repository.

Exit status

0
Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the `spectrumscale` command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For more information, see *Requirements for administering a GPFS(tm) file system* in *IBM Spectrum Scale: Administration Guide*.

Examples

Creating a new IBM Spectrum Scale cluster

1. To designate your installer node, issue this command:

```
spectrumscale setup -s 192.168.0.1
```

2. To designate NSD server nodes in your environment to use for the installation, issue this command:

```
./spectrumscale node add FQDN -n
```

3. To add four non-shared NSDs seen by a primary NSD server only, issue this command:

```
./spectrumscale nsd add -p FQDN_of_Primary_NSD_Server /dev/dm-1 /dev/dm-2 /dev/dm-3 /dev/
dm-4
```

4. To add four non-shared NSDs seen by both a primary NSD server and a secondary NSD server, issue this command:

```
./spectrumscale nsd add -p FQDN_of_Primary_NSD_Server -s FQDN_of_Secondary_NSD_Server
 /dev/dm-1 /dev/dm-2 /dev/dm-3 /dev/dm-4
```

5. To define a shared root file system using two NSDs and a file system `fs1` using two NSDs, issue these commands:

```
./spectrumscale nsd list
./spectrumscale filesystem list
./spectrumscale nsd modify nsd1 -fs cesSharedRoot
./spectrumscale nsd modify nsd2 -fs cesSharedRoot
./spectrumscale nsd modify nsd3 -fs fs1
./spectrumscale nsd modify nsd4 -fs fs1
```

6. To designate GUI nodes in your environment to use for the installation, issue this command:

```
./spectrumscale node add FQDN -g -a
```

7. To designate additional client nodes in your environment to use for the installation, issue this command:

```
./spectrumscale node add FQDN
```

8. To allow the installation toolkit to reconfigure Performance Monitoring if it detects any existing configurations, issue this command:

```
./spectrumscale config perfmon -r on
```

9. To name your cluster, issue this command:

```
./spectrumscale config gpfs -c Cluster_Name
```

10. To configure the call home function with the mandatory parameters, issue this command:

```
./spectrumscale callhome config -n username -i 456123 -e username@example.com -cn US
```

If you do not want to use call home, disable it by issuing the following command:

```
./spectrumscale callhome disable
```

For more information, see *Enabling and configuring call home using the installation toolkit* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

11. To review the configuration prior to installation, issue these commands:

```
./spectrumscale node list
./spectrumscale nsd list
./spectrumscale filesystem list
./spectrumscale config gpfs --list
```

12. To start the installation on your defined environment, issue these commands:

```
./spectrumscale install --precheck
./spectrumscale install
```

Deploying protocols on an existing cluster

Note: If your cluster contains ESS, see the *Adding protocols to a cluster containing ESS* section.

1. To designate your installer node, issue this command:

```
spectrumscale setup -s 192.168.0.1
```

2. To designate protocol nodes in your environment to use for the deployment, issue this command:

```
./spectrumscale node add FQDN -p
```

3. To designate GUI nodes in your environment to use for the deployment, issue this command:

```
./spectrumscale node add FQDN -g -a
```

4. To configure protocols to point to a file system that will be used as a shared root, issue this command:

```
./spectrumscale config protocols -f FS_Name -m Shared_FS_Mountpoint_Or_Path
```

5. To configure a pool of export IPs, issue this command:

```
./spectrumscale config protocols -e Comma_Separated_List_of_Exportpool_IPs
```

6. If you are using the CES interface mode, to set the network interfaces, issue this command:

```
./spectrumscale config protocols -i INTERFACE1,INTERFACE2,...
```

7. To enable NFS on all protocol nodes, issue this command:

```
./spectrumscale enable nfs
```

8. To enable SMB on all protocol nodes, issue this command:

```
./spectrumscale enable smb
```

9. To enable object on all protocol nodes, issue these commands:

```
./spectrumscale enable object
./spectrumscale config object -au Admin_User -ap Admin_Password -dp Database_Password
./spectrumscale config object -e FQDN
./spectrumscale config object -f FS_Name -m FS_Mountpoint
./spectrumscale config object -o Object_Fileset
```

10. To enable file audit logging, issue the following command:

```
./spectrumscale fileauditlogging enable
```

For more information, see *Enabling and configuring file audit logging using the installation toolkit* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

11. To review the configuration prior to deployment, issue these commands:

```
./spectrumscale config protocols
./spectrumscale config object
./spectrumscale node list
```

12. To deploy protocols on your defined environment, issue these commands:

```
./spectrumscale deploy --precheck
./spectrumscale deploy
```

Upgrading an IBM Spectrum Scale cluster

1. Extract the IBM Spectrum Scale package for the required code level by issuing a command similar to the following depending on the package name:

```
./Spectrum_Scale_Standard-5.1.x.x-xxxxx
```

2. Populate the cluster definition file with the current cluster state by issuing the following command:

```
./spectrumscale config populate
```

3. **[Optional]** Enable the prompt to users to shut down their workloads before starting the upgrade by issuing the following command.

```
./spectrumscale upgrade config workload -p on
```

4. Run the upgrade precheck from the installer directory of the latest code level extraction by issuing commands similar to the following:

```
cd /usr/lpp/mmfs/Latest_Code_Level_Directory/ansible-toolkit
./spectrumscale upgrade precheck
```

5. **[Optional]** Specify nodes as offline by issuing the following command, if services running on these nodes are stopped or down.

```
./spectrumscale upgrade config offline -N Node
```

6. **[Optional]** Exclude nodes that you do not want to upgrade at this point by issuing the following command.

```
./spectrumscale upgrade config exclude -N Node
```

7. Run the upgrade by issuing this command:

```
cd /usr/lpp/mmfs/Latest_Code_Level_Directory/ansible-toolkit
./spectrumscale upgrade run
```

Adding to an installation process

1. To add nodes to an installation, do the following:

   a. Add one or more node types using the following commands:

- Client nodes:

```
./spectrumscale node add FQDN
```

- NSD nodes:

```
./spectrumscale node add FQDN -n
```

- Protocol nodes:

```
./spectrumscale node add FQDN -p
```

- GUI nodes:

```
./spectrumscale node add FQDN -g -a
```

b. Install GPFS on the new nodes using the following commands:

```
./spectrumscale install --precheck
./spectrumscale install
```

c. If protocol nodes are being added, deploy protocols using the following commands:

```
./spectrumscale deploy --precheck
./spectrumscale deploy
```

2. To add NSDs to an installation, do the following:

a. Verify that the NSD server connecting this new disk runs an OS compatible with the installation toolkit and that the NSD server exists within the cluster.

b. Add NSDs to the installation using the following command:

```
./spectrumscale nsd add -p FQDN_of_Primary_NSD_Server Path_to_Disk_Device_File
```

c. Run the installation using the following commands:

```
./spectrumscale install --precheck
./spectrumscale install
```

3. To add file systems to an installation, do the following:

a. Verify that free NSDs exist and that they can be listed by the installation toolkit using the following commands.

```
mmlsnsd
./spectrumscale nsd list
```

b. Define the file system using the following command:

```
./spectrumscale nsd modify NSD -fs File_System_Name
```

c. Add the file system by using the following commands:

```
./spectrumscale install --precheck
./spectrumscale install
```

4. To enable another protocol on an existing cluster that has protocols enabled, do the following steps depending on your configuration:

a. Enable NFS on all protocol nodes using the following command:

```
./spectrumscale enable nfs
```

b. Enable SMB on all protocol nodes using the following command:

```
./spectrumscale enable smb
```

c. Enable object on all protocol nodes using the following commands:

```
./spectrumscale enable object
./spectrumscale config object -au Admin_User -ap Admin_Password -dp Database_Password
./spectrumscale config object -e FQDN
./spectrumscale config object -f FS_Name -m FS_Mountpoint
./spectrumscale config object -o Object_Fileset
```

d. Enable the new protocol using the following commands:

```
./spectrumscale deploy --precheck
./spectrumscale deploy
```

Using the installation toolkit in cluster containing ESS

For a list of assumptions and restrictions for using the installation toolkit in a cluster containing ESS, see the -st SetupType option. When using the installation toolkit in a cluster containing ESS, use the following high-level steps.

1. Add protocol nodes in the ESS cluster by issuing the following command.

   ```
   ./spectrumscale node add NodeName -p
   ```

   You can add other types of nodes such as client nodes, NSD servers, and so on depending on your requirements. For more information, see *Defining configuration options for the installation toolkit* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.
2. Specify one of the newly added protocol nodes as the installer node and specify the setup type as ess by issuing the following command.

   ```
   ./spectrumscale setup -s NodeIP -i SSHIdentity -st ess
   ```

   The installer node is the node on which the installation toolkit is extracted and from where the installation toolkit command, **spectrumscale**, is initiated.
3. Specify the EMS node of the ESS system to the installation toolkit by issuing the following command.

   ```
   ./spectrumscale node add NodeName -e
   ```

   This node is also automatically specified as the admin node. The admin node, which must be the EMS node in an ESS configuration, is the node that has access to all other nodes to perform configuration during the installation.
4. Proceed with specifying other configuration options, installing, and deploying by using the installation toolkit. For more information, see *Defining configuration options for the installation toolkit*, *Installing GPFS and creating a GPFS cluster*, and *Deploying protocols* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

For more information, see *ESS awareness with the installation toolkit* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Manually adding protocols to a cluster containing ESS

For information on preparing a cluster that contains ESS for adding protocols, see *Preparing a cluster that contains ESS for adding protocols* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

After you have prepared your cluster that contains ESS for adding protocols, you can use commands similar to the ones listed in the *Deploying protocols on an existing cluster* section.

Using the installation toolkit in an IBM Spectrum Scale Erasure Code Edition environment

- Specify the installer node and the setup type as ece in the cluster definition file for IBM Spectrum Scale Erasure Code Edition.

  ```
  ./spectrumscale setup -s InstallerNodeIP -st ece
  ```

- Add scale-out nodes for IBM Spectrum Scale Erasure Code Edition in the cluster definition file.

```
./spectrumscale node add NodeName -so
```

- Define the recovery group for IBM Spectrum Scale Erasure Code Edition in the cluster definition file.

```
./spectrumscale recoverygroup define -N Node1,Node2,...,NodeN
```

- Define vdisk sets for IBM Spectrum Scale Erasure Code Edition in the cluster definition file.

```
./spectrumscale vdiskset define -rg RgName -code RaidCode -bs BlockSize -ss SetSize
```

- Define the file system for IBM Spectrum Scale Erasure Code Edition in the cluster definition file.

```
./spectrumscale filesystem define -fs FileSystem -vs VdiskSet
```

Creating a new HDFS cluster by using the installation toolkit

1. Enable HDFS in the cluster definition file.

```
./spectrumscale enable hdfs
```

2. Run the installation precheck procedure.

```
./spectrumscale install --precheck
```

3. Run the installation procedure.

```
./spectrumscale install
```

4. Set the CES IP addresses.

```
./spectrumscale config protocols -e CES_IP1,CES_IP2,...,CES_IPN
```

5. If you are using the CES interface mode, set the network interfaces.

```
./spectrumscale config protocols -i INTERFACE1,INTERFACE2,...
```

6. Set up the CES shared root file system.

```
./spectrumscale config protocols -f cesSharedRoot -m /ibm/cesSharedRoot
```

7. Define the properties for the new HDFS cluster.

```
./spectrumscale config hdfs new -n hdfscluster1 -nn namenode1,namenode2
-dn datanode1,datanode2 -f fs1 -d DataDir
```

8. Run the deployment precheck procedure.

```
./spectrumscale deploy --precheck
```

9. Run the deployment procedure.

```
./spectrumscale deploy
```

Adding name nodes or data nodes in an existing HDFS cluster by using the installation toolkit

1. Define the properties for the new HDFS cluster.

```
./spectrumscale config hdfs add -n hdfscluster2 -nn namenode3 -dn datanode3
```

2. Run the deployment precheck procedure.

```
./spectrumscale deploy --precheck
```

3. Run the deployment procedure.

```
./spectrumscale deploy
```

Importing an existing HDFS configuration by using the installation toolkit

1. Define the properties for the new HDFS cluster.

   ```
   ./spectrumscale config hdfs import -l hdfs_config_dir
   ```

2. Run the deployment precheck procedure.

   ```
   ./spectrumscale deploy --precheck
   ```

3. Run the deployment procedure.

   ```
   ./spectrumscale deploy
   ```

Diagnosing an error during install, deploy, or upgrade

1. Note the screen output indicating the error. This helps in narrowing down the general failure.

   When a failure occurs, the screen output also shows the log file containing the failure.
2. Open the log file in an editor such as `vi`.
3. Go to the end of the log file and search upwards for the text FATAL.
4. Find the topmost occurrence of FATAL (or first FATAL error that occurred) and look above and below this error for further indications of the failure.

For more information, see *Finding deployment related error messages more easily and using them for failure analysis* in *IBM Spectrum Scale: Problem Determination Guide*.

See also

- *Installing IBM Spectrum Scale on Linux nodes and deploying protocols* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.
- *Configuring with the spectrumscale installation toolkit* in *IBM Spectrum Scale: Administration Guide*.
- “mmchconfig command” on page 170
- “mmlscluster command” on page 492
- “mmlsconfig command” on page 495
- “mmnfs command” on page 560
- “mmobj command” on page 573
- “mmsmb command” on page 709
- “mmuserauth command” on page 738

Location

`/usr/lpp/mmfs/5.1.1.x/ansible-toolkit`

Chapter 2. IBM Spectrum Scale Data Management API for GPFS information

The Data Management Application Programming Interface (DMAPI) for (GPFS) is based on The Open Group's System Management: Data Storage Management (XDSM) API Common Applications Environment (CAE) Specification C429, The Open Group, ISBN 1-85912-190-X specification. The implementation is compliant with the standard. Some optional features are not implemented.

The XDSM DMAPI model is intended mainly for a single-node environment. Some of the key concepts, such as sessions, event delivery, and recovery, required enhancements for a multiple-node environment such as GPFS.

Overview of IBM Spectrum Scale Data Management API for GPFS

The Data Management Application Programming Interface (DMAPI) for GPFS allows you to monitor events associated with a GPFS file system or with an individual file. You can also manage and maintain file system data.

See the IBM Spectrum Scale FAQ in IBM Documentation for the current limitations of DMAPI-managed file systems.

Note: IBM Spectrum Protect for Space Management for GPFS file systems is not available for Windows.

DMAPI for GPFS is compliant with the Open Group's XDSM Standard and includes these features:

- "GPFS-specific DMAPI events" on page 801
- "DMAPI functions" on page 803
- "DMAPI configuration attributes" on page 806
- "DMAPI restrictions for GPFS" on page 807

GPFS-specific DMAPI events

There are three GPFS-specific DMAPI events: events implemented in DMAPI for GPFS, optional events that are not implemented in DMAPI for GPFS, and GPFS-specific attribute events that are not part of the DMAPI standard.

For more information, see:

- "Events implemented in DMAPI for GPFS" on page 801
- "Optional events that are not implemented in DMAPI for GPFS" on page 802
- "GPFS-specific attribute events that are not part of the DMAPI standard" on page 802

Events implemented in DMAPI for GPFS

These are the events, as defined in the *System Management: Data Storage Management (XDSM) API* Common Applications Environment (CAE) Specification C429, The Open Group, ISBN 1-85912-190-X, implemented in DMAPI for GPFS:

File system administration events

- mount
- preunmount
- unmount
- nospace

Namespace events

- create, postcreate
- remove, postremove
- rename, postrename
- symlink, postsymlink
- link, postlink

Data events

- read
- write
- truncate

Metadata events

- attribute
- destroy
- close

Pseudo event

- user event

GPFS guarantees that asynchronous events are delivered, except when the GPFS daemon fails. Events are enqueued to the session before the corresponding file operation completes. For further information on failures, see "Failure and recovery of IBM Spectrum Scale Data Management API for GPFS" on page 837.

Optional events that are not implemented in DMAPI for GPFS

The following optional events, as defined in the *System Management: Data Storage Management (XDSM) API* Common Applications Environment (CAE) Specification C429, The Open Group, ISBN 1-85912-190-X, are not implemented in DMAPI for GPFS:

File system administration event

- debut

Metadata event

- cancel

GPFS-specific attribute events that are not part of the DMAPI standard

GPFS generates the following attribute events for DMAPI that are specific to GPFS and not part of the DMAPI standard:

- Pre-permission change
- Post-permission change

For additional information, refer to "GPFS-specific DMAPI events" on page 834.

DMAPI functions

All mandatory DMAPI functions and most optional functions that are defined in the *System Management: Data Storage Management (XDSM) API* Common Applications Environment (CAE) Specification C429, The Open Group, ISBN 1-85912-190-X, are implemented in DMAPI for GPFS.

For C declarations of all the functions implemented in DMAPI for GPFS, refer to the `dmapi.h` file located in the **/usr/lpp/mmfs/include** directory.

For changes and restrictions on functions in DMAPI for GPFS, see "Usage restrictions on DMAPI functions" on page 818, and "Semantic changes to DMAPI functions" on page 833.

Mandatory functions implemented in DMAPI for GPFS

These mandatory functions, as defined in the *System Management: Data Storage Management (XDSM) API* Common Applications Environment (CAE) Specification C429, The Open Group, ISBN 1-85912-190-X, are implemented in DMAPI for GPFS.

For C declarations of all the mandatory functions implemented in DMAPI for GPFS, refer to the `dmapi.h` file located in the **/usr/lpp/mmfs/include** directory. However, for a quick description of the mandatory functions and their applications, refer to the following set of functions:

dm_create_session
Create a new session.

dm_create_userevent
Create a pseudo-event message for a user.

dm_destroy_session
Destroy an existing session.

dm_fd_to_handle
Create a file handle using a file descriptor.

dm_find_eventmsg
Return the message for an event.

dm_get_allocinfo
Get a file's current allocation information.

dm_get_bulkattr
Get bulk attributes of a file system.

dm_get_config
Get specific data on DMAPI implementation.

dm_get_config_events
List all events supported by the DMAPI implementation.

dm_get_dirattrs
Return a directory's bulk attributes.

dm_get_eventlist
Return a list of an object's enabled events.

dm_get_events
Return the next available event messages.

dm_get_fileattr
Get file attributes.

dm_get_mountinfo
Return details from a mount event.

dm_get_region
Get a file's managed regions.

dm_getall_disp
For a given session, return the disposition of all file system's events.

dm_getall_sessions
Return all extant sessions.

dm_getall_tokens
Return a session's outstanding tokens.

dm_handle_cmp
Compare file handles.

dm_handle_free
Free a handle's storage.

dm_handle_hash
Hash the contents of a handle.

dm_handle_is_valid
Check a handle's validity.

dm_handle_to_fshandle
Return the file system handle associated with an object handle.

dm_handle_to_path
Return a path name from a file system handle.

dm_init_attrloc
Initialize a bulk attribute location offset.

dm_init_service
Initialization processing that is implementation-specific.

dm_move_event
Move an event from one session to another.

dm_path_to_fshandle
Create a file system handle using a path name.

dm_path_to_handle
Create a file handle using a path name.

dm_query_right
Determine an object's access rights.

dm_query_session
Query a session.

dm_read_invis
Read a file without using DMAPI events.

dm_release_right
Release an object's access rights.

dm_request_right
Request an object's access rights.

dm_respond_event
Issue a response to an event.

dm_send_msg
Send a message to a session.

dm_set_disp
For a given session, set the disposition of all file system's events.

dm_set_eventlist
For a given object, set the list of events to be enabled.

dm_set_fileattr
Set a file's time stamps, ownership and mode.

dm_set_region
Set a file's managed regions.

dm_write_invis
Write to a file without using DMAPI events.

Optional functions implemented in DMAPI for GPFS

These optional functions, as defined in the *System Management: Data Storage Management (XDSM) API* Common Applications Environment (CAE) Specification C429, The Open Group, ISBN 1-85912-190-X, are implemented in DMAPI for GPFS.

For C declarations of these optional functions implemented in DMAPI for GPFS, refer to the `dmapi.h` file located in the `/usr/lpp/mmfs/include` directory. However, for a quick description of the optional functions and their applications, refer to the following set of functions:

dm_downgrade_right
Change an exclusive access right to a shared access right.

dm_get_bulkall
Return a file system's bulk data management attributes.

dm_get_dmattr
Return a data management attribute.

dm_getall_dmattr
Return all data management attributes of a file.

dm_handle_to_fsid
Get a file system ID using its handle.

dm_handle_to_igen
Get inode generation count using a handle.

dm_handle_to_ino
Get inode from a handle.

dm_make_handle
Create a DMAPI object handle.

dm_make_fshandle
Create a DMAPI file system handle.

dm_punch_hole
Make a hole in a file.

dm_probe_hole
Calculate the rounded result of the area where it is assumed that a hole is to be punched.

dm_remove_dmattr
Delete a data management attribute.

dm_set_dmattr
Define or update a data management attribute.

dm_set_return_on_destroy
Indicate a DM attribute to return with destroy events.

dm_sync_by_handle
Synchronize the in-memory state of a file with the physical medium.

dm_upgrade_right
Change a currently held access right to be exclusive.

Optional functions that are not implemented in DMAPI for GPFS

There are optional functions that are not implemented in DMAPI for GPFS.

The following optional functions, as defined in the *System Management: Data Storage Management (XDSM) API* Common Applications Environment (CAE) Specification C429, The Open Group, ISBN 1-85912-190-X, are `not` implemented in DMAPI for GPFS:

dm_clear_inherit
Reset the inherit-on-create status of an attribute.

dm_create_by_handle
Define a file system object using a DM handle.

dm_getall_inherit
Return a file system's inheritable attributes.

dm_mkdir_by_handle
Define a directory object using a handle.

dm_obj_ref_hold
Put a hold on a file system object.

dm_obj_ref_query
Determine if there is a hold on a file system object.

dm_obj_ref_rele
Release the hold on a file system object.

dm_pending
Notify FS of slow DM application processing.

dm_set_inherit
Indicate that an attribute is inheritable.

dm_symlink_by_handle
Define a symbolic link using a DM handle.

GPFS-specific DMAPI functions

There are several GPFS-specific DMAPI functions that are not part of the DMAPI open standard.

The GPFS-specific functions are listed and described in "Definitions for GPFS-specific DMAPI functions" on page 820.

DMAPI configuration attributes

The *System Management: Data Storage Management (XDSM) API* Common Applications Environment (CAE) Specification C429, The Open Group, ISBN 1-85912-190-X defines a set of configuration attributes to be exported by each DMAPI implementation. These attributes specify which optional features are supported and give bounds on various resources.

The Data Management (DM) application can query the attribute values using the function `dm_get_config`. It can also query which events are supported, using the function `dm_get_config_events`.

The functions `dm_get_config` and `dm_get_config_events` receive a file handle from input arguments *hanp* and *hlen*. In GPFS, both functions ignore the handle, as the configuration is not dependent on the specific file or file system. This enables the DM application to query the configuration during initialization, when file handles may not yet be available.

Note: To ensure that the most current values are being used, the DM application should always query the configuration at runtime by using `dm_get_config`.

Table 33 on page 806 shows the attribute values that are used in DMAPI for GPFS:

Table 33. DMAPI configuration attributes

Name	Value
DM_CONFIG_BULKALL	1
DM_CONFIG_CREATE_BY_HANDLE	0
DM_CONFIG_DTIME_OVERLOAD	1
DM_CONFIG_LEGACY	1
DM_CONFIG_LOCK_UPGRADE	1
DM_CONFIG_MAX_ATTR_ON_DESTROY	1022

Table 33. DMAPI configuration attributes (continued)	
Name	**Value**
DM_CONFIG_MAX_ATTRIBUTE_SIZE	1022
DM_CONFIG_MAX_HANDLE_SIZE	32
DM_CONFIG_MAX_MANAGED_REGIONS	32
DM_CONFIG_MAX_MESSAGE_DATA	4096
DM_CONFIG_OBJ_REF	0
DM_CONFIG_PENDING	0
DM_CONFIG_PERS_ATTRIBUTE	1
DM_CONFIG_PERS_EVENTS	1
DM_CONFIG_PERS_INHERIT_ATTRIBS	0
DM_CONFIG_PERS_MANAGED_REGIONS	1
DM_CONFIG_PUNCH_HOLE	1
DM_CONFIG_TOTAL_ATTRIBUTE_SPACE	7168
DM_CONFIG_WILL_RETRY	0

Attribute value DM_CONFIG_TOTAL_ATTRIBUTE_SPACE is per file. The entire space is available for opaque attributes. Non-opaque attributes (event list and managed regions) use separate space.

DMAPI restrictions for GPFS

All DMAPI APIs must be called from nodes that are in the cluster where the file system is created. DMAPI APIs may **not** be invoked from a remote cluster.

Furthermore, GPFS also places the following DMAPI API restrictions:

- Running dm_get_events with the DM_EV_WAIT flag set causes the calling process to wait uninterruptibly.
- Interacting with a handle after calling dm_handle_free will result in undefined behavior.

In addition to the DMAPI API restrictions, GPFS places the following restrictions on the use of file system snapshots when you have DMAPI enabled:

- Snapshots cannot coexist with file systems using GPFS 3.1 or earlier.
- GPFS 3.2 and later permits snapshots with DMAPI-enabled file systems. However, GPFS places the following restrictions on DMAPI access to the snapshot files:
  - The DM server may read files in a snapshot using dm_read_invis.
  - The DM server is not allowed to modify or delete the file using dm_write_invis or dm_punch_hole.
  - The DM server is not allowed to establish a managed region on the file.
  - Snapshot creation or deletion does not generate DMAPI namespace events.
  - Snapshots of a file are not managed regardless of the state of the original file and they do not inherit the DMAPI attributes of the original file.

Concepts of IBM Spectrum Scale Data Management API for GPFS

The XDSM standard is intended mainly for a single-node environment. Some of the key concepts in the standard such as sessions, event delivery, mount and unmount, and failure and recovery, are not well defined for a multiple-node environment such as GPFS.

For a list of restrictions and coexistence considerations, see "Usage restrictions on DMAPI functions" on page 818.

All DMAPI APIs must be called from nodes that are in the cluster where the file system is created.

Key concepts of DMAPI for GPFS include these areas:

Sessions

In GPFS, a session is associated only with the node on which the session was created. This node is known as the *session node*.

Events are generated at any node where the file system is mounted. The node on which a given event is generated is called the *source node* of that event. The event is delivered to a session queue on the session node.

There are restrictions as to which DMAPI functions can and cannot be called from a node other than the session node. In general, functions that change the state of a session or event can only be called on the session node. For example, the maximum number of DMAPI sessions that can be created on a node is 4000. See "Usage restrictions on DMAPI functions" on page 818 for details.

Session ids are unique over time within a GPFS cluster. When an existing session is assumed, using `dm_create_session`, the new session id returned is the same as the old session id.

A session fails when the GPFS daemon fails on the session node. Unless this is a total failure of GPFS on all nodes, the session is recoverable. The DM application is expected to assume the old session, possibly on another node. This will trigger the reconstruction of the session queue. All pending synchronous events from surviving nodes are resubmitted to the recovered session queue. Such events will have the same token id as before the failure, except mount events. Asynchronous events, on the other hand, are lost when the session fails. See "Failure and recovery of IBM Spectrum Scale Data Management API for GPFS" on page 837 for information on failure and recovery.

Data management events

Data management events arrive on a session queue from any of the nodes in the GPFS cluster.

The source node of the event is identified by the `ev_nodeid` field in the header of each event message in the structure `dm_eventmsg`. The identification is the GPFS cluster data node number, which is attribute `node_number` in the `mmsdrfs2` file for a PSSP node or `mmsdrfs` file for any other type of node.

Data Management events are generated only if the following two conditions are true:

1. The event is enabled.
2. It has a disposition.

A file operation will fail with the **EIO** error if there is no disposition for an event that is enabled and would otherwise be generated.

A list of enabled events can be associated individually with a file and globally with an entire file system. The XDSM standard leaves undefined the situation where the individual and the global event lists are in conflict. In GPFS, such conflicts are resolved by always using the individual event list, if it exists.

Note: The XDSM standard does not provide the means to remove the individual event list of a file. Thus, there is no way to enable or disable an event for an entire file system without explicitly changing each conflicting individual event list.

In GPFS, event lists are persistent.

Event dispositions are specified per file system and are not persistent. They must be set explicitly after the session is created.

Event generation mechanisms have limited capacity. In case resources are exceeded, new file operations will wait indefinitely for free resources.

File operations wait indefinitely for a response from synchronous events. The `dmapiEventTimeout` configuration attribute on the `mmchconfig` command, can be used to set a timeout on events that originate from NFS file operations. This is necessary because NFS servers have a limited number of threads that cannot be blocked for long periods of time. Refer to "GPFS configuration attributes for DMAPI" on page 814 and "Support for NFS" on page 812.

The XDSM standard permits asynchronous events to be discarded at any time. In GPFS, asynchronous events are guaranteed when the system runs normally, but may be lost during abnormal conditions, such as failure of GPFS on the session node. Asynchronous events are delivered in a timely manner. That is, an asynchronous event is enqueued to the session before the corresponding file operation completes.

Figure 1 on page 809 shows the flow of a typical synchronous event in a multiple-node GPFS environment. The numbered arrows in the figure correspond to the following steps:

1. The user application on the source node performs a file operation on a GPFS file. The file operation thread generates a synchronous event and blocks, waiting for a response.
2. GPFS on the source node sends the event to GPFS on the session node, according to the disposition for that event. The event is enqueued to the session queue on the session node.
3. The Data Management application on the session node receives the event (using `dm_get_events`) and handles it.
4. The Data Management application on the session node responds to the event (using `dm_respond_event`).
5. GPFS on the session node sends the response to GPFS on the source node.
6. GPFS on the source node passes the response to the file operation thread and unblocks it. The file operation continues.

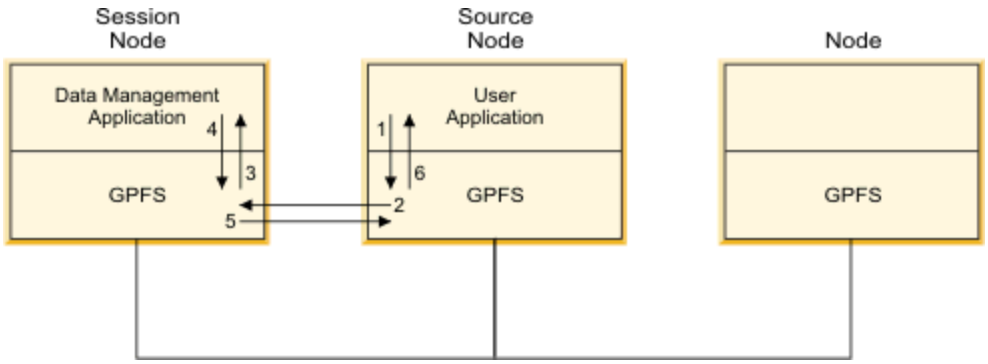

Figure 1. Flow of a typical synchronous event in a multiple-node GPFS environment

Reliable DMAPI destroy events

A metadata destroy event is generated when the operating system has destroyed an object. This type of event is different from a remove event, which is a namespace event and is not related to the destruction of an object. A reliable destroy event supports synchronous destroy events in the same way that other synchronous events do. When a synchronous event is generated, a user process is suspended in the kernel; it will be suspended until a DM application issues an explicit response to the event. The DM application at the session that supports the reliable destroy event must be capable of handling the synchronous destroy event. In other words, it must respond to the DM_EVENT_DESTROY event with DM_RESPOND_EVENT. Otherwise, the event will wait forever at the session node for a response. Based on this, it is recommended that the cluster not be made up of nodes that are running back-level code and new code, because the destroy event is not reliable in a mixed environment.

Mount and unmount

The XDSM standard implicitly assumes that there is a single mount, pre-unmount and unmount event per file system. In GPFS, a separate mount event is generated by each mount operation on each node. Similarly, if the pre-unmount and unmount events are enabled, they are generated by each unmount operation on each node. Thus, there may be multiple such events for the same file system.

To provide additional information to the DM application, the mode field in the respective event message structures (me_mode for mount, and ne_mode for pre-unmount and unmount) has a new flag, DM_LOCAL_MOUNT, which is not defined in the standard. When the flag is set, the mount or unmount operation is local to the session node. In addition, the new field ev_nodeid in the header of the event message can be used to identify the source node where the mount or unmount operation was invoked. The identification is the GPFS cluster data node number, which is attribute node_number in the mmsdrfs2 file for a PSSP node or mmsdrfs file for any other type of node.

The mount event is sent to multiple sessions that have a disposition for it. If there is no disposition for the mount event, the mount operation fails with an **EIO** error.

There is no practical way to designate the *last* unmount, since there is no serialization of all mount and unmount operations of each file system. Receiving an unmount event with the value 0 in the ne_retcode field is no indication that there will be no further events from the file system.

An unmount initiated internally by the GPFS daemon, due to file system forced unmount or daemon shutdown, will not generate any events. Consequently, there need not be a match between the number of mount events and the number of pre-unmount or unmount events for a given file system.

The dmapiMountTimeout attribute on the mmchconfig command enables blocking the mount operation for a limited time until some session has set the mount disposition. This helps GPFS and the DM application synchronize during initialization. See "GPFS configuration attributes for DMAPI" on page 814 and "Initializing the Data Management application" on page 816.

Mount events are enqueued on the session queue ahead of any other events. This gives mount events a higher priority, which improves the response time for mount events when the queue is very busy.

If the DM_UNMOUNT_FORCE flag is set in the pre-unmount event message, the response of the DM application to the pre-unmount event is ignored, and the forced unmount proceeds. If the DM_LOCAL_MOUNT flag is also set, the forced unmount will result in the loss of all access rights of the given file system that are associated with any local session.

If the unmount is not forced (the DM_UNMOUNT_FORCE flag is not set), and the DM_LOCAL_MOUNT flag is set, the DM application is expected to release all access rights on files of the given file system associated with any local session. If any access rights remain held after the DM_RESP_CONTINUE response is given, the unmount will fail with **EBUSY**. This is because access rights render the file system busy, similar to other locks on files.

The function dm_get_mountinfo can be called from any node, even if the file system is not mounted on that node. The dm_mount_event structure returned by the dm_get_mountinfo function provides the following enhanced information. The me_mode field contains two new flags, DM_LOCAL_MOUNT and DM_REMOTE_MOUNT. At least one of the two flags is always set. When both flags are set simultaneously, it

is an indication that the file system is mounted on the local node, as well as one or more other (remote) nodes. When only DM_LOCAL_MOUNT is set, it is an indication that the file system is mounted on the local node but not on any other node. When only DM_REMOTE_MOUNT is set, it is an indication that the file system is mounted on some remote node, but not on the local node.

In the latter case (only DM_REMOTE_MOUNT is set), the fields me_roothandle and me_handle2 (the mount point handle) in the dm_mount_event structure are set to DM_INVALID_HANDLE. Also in this case, the me_name1 field (the mount point path) is taken from the stanza in the file /etc/filesystems on one of the remote nodes (with the use of GPFS cluster data, the stanzas on all nodes are identical).

The enhanced information provided by the dm_get_mountinfo function can be useful during the processing of mount and pre-unmount events. For example, before responding to a mount event from a remote (non-session) node, dm_get_mountinfo could be invoked to find out whether the file system is already mounted locally at the session node, and if not, initiate a local mount. On receiving a pre-unmount event from the local session node, it is possible to find out whether the file system is still mounted elsewhere, and if so, fail the local unmount or delay the response until after all remote nodes have unmounted the file system.

Note: The DM_REMOTE_MOUNT flag is redundant in the dm_mount_event structure obtained from the mount event (as opposed to the dm_get_mountinfo function).

Tokens and access rights

A DMAPI token is an identifier of an outstanding event (a synchronous event that the DM application has received and is currently handling). The token is unique over time in the cluster. The token becomes invalid when the event receives a response.

The main purpose of tokens is to convey access rights in DMAPI functions. Access rights are associated with a specific event token. A function requiring access rights to some file may present an event token that has the proper access rights.

DMAPI functions can also be invoked using DM_NO_TOKEN, in which case sufficient access protection is provided for the duration of the operation. This is semantically equivalent to holding an access right, but no access right on the file is actually acquired.

In GPFS, when an event is received, its token has no associated access rights.

DM access rights are implemented in GPFS using an internal lock on the file. Access rights can be acquired, changed, queried, and released only at the session node. This is an implementation restriction caused by the GPFS locking mechanisms.

In GPFS, it is not possible to set an access right on an entire file system from the file system handle. Thus, DMAPI function calls that reference a file system, using a file system handle, are not allowed to present a token and must specify DM_NO_TOKEN. For the same reason, functions that acquire or change access rights are not allowed to present a file system handle.

Holding access rights renders the corresponding file system busy at the session node, preventing normal (non-forced) unmount. This behavior is similar to that of other locks on files. When receiving a pre-unmount event, the DM application is expected to release all access rights before responding. Otherwise, the unmount operation will fail with an **EBUSY** error.

All access rights associated with an event token are released when the response is given. There is no transfer of access rights from DMAPI to the file operation thread. The file operation will acquire any necessary locks after receiving the response of the event.

Parallelism in Data Management applications

Given the multiple-node environment of GPFS, it is desirable to exploit parallelism in the Data Management application as well.

This can be accomplished in several ways:

- On a given session node, multiple DM application threads can access the same file in parallel, using the same session. There is no limit on the number of threads that can invoke DMAPI functions simultaneously on each node.
- Multiple sessions, each with event dispositions for a different file system, can be created on separate nodes. Thus, files in different file systems can be accessed independently and simultaneously, from different session nodes.
- Dispositions for events of the same file system can be partitioned among multiple sessions, each on a different node. This distributes the management of one file system among several session nodes.
- Although GPFS routes all events to a single session node, data movement may occur on multiple nodes. The function calls `dm_read_invis`, `dm_write_invis`, `dm_probe_hole`, and `dm_punch_hole` are honored from a root process on another node, provided it presents a session ID for an established session on the session node.

  A DM application may create a *worker process*, which exists on any node within the GPFS cluster. This worker process can move data to or from GPFS using the `dm_read_invis` and `dm_write_invis` functions. The worker processes must adhere to these guidelines:

  1. They must run as root.
  2. They must present a valid session ID that was obtained on the session node.
  3. All writes to the same file which are done in parallel must be done in multiples of the file system block size, to allow correct management of disk blocks on the writes.
  4. No DMAPI calls other than `dm_read_invis`, `dm_write_invis`, `dm_probe_hole`, and `dm_punch_hole` may be issued on nodes other than the session node. This means that any rights required on a file must be obtained within the session on the session node, prior to the data movement.
  5. There is no persistent state on the nodes hosting the worker process. It is the responsibility of the DM application to recover any failure which results from the failure of GPFS or the data movement process.

Data Management attributes

Data Management attributes can be associated with any individual file. There are opaque and non-opaque attributes.

An opaque attribute has a unique name, and a byte string value which is not interpreted by the DMAPI implementation. Non-opaque attributes, such as managed regions and event lists, are used internally by the DMAPI implementation.

DM attributes are persistent. They are kept in a hidden file in the file system.

GPFS provides two *quick access* single-bit opaque DM attributes for each file, stored directly in the inode. These attributes are accessible through regular DMAPI functions, by specifying the reserved attribute names `_GPFSQA1` and `_GPFSQA2` (where `_GPF` is a reserved prefix). The attribute data must be a single byte with contents 0 or 1.

Support for NFS

A DM application could be slow in handling events. NFS servers have a limited number of threads which must not all be blocked simultaneously for extended periods of time. GPFS provides a mechanism to guarantee progress of NFS file operations that generate data events without blocking the server threads indefinitely.

The mechanism uses a timeout on synchronous events. Initially the NFS server thread is blocked on the event. When the timeout expires, the thread unblocks and the file operation fails with an `ENOTREADY` error code. The event itself continues to exist and will eventually be handled. When a response for the event arrives at the source node it is saved. NFS is expected to periodically retry the operation. The retry will either find the response which has arrived between retries, or cause the operation to fail again with `ENOTREADY`. After repeated retries, the operation is eventually expected to succeed.

The interval is configurable using the `dmapiEventTimeout` configuration attribute on the `mmchconfig` command. See "GPFS configuration attributes for DMAPI" on page 814. The default is no timeout.

The timeout mechanism is activated only for data events (read, write, truncate), and only when the file operation comes from NFS.

For the parameter change to take effect, restart the GPFS daemon on all nodes.

Quota

GPFS supports user quota. When `dm_punch_hole` is invoked, the file owner's quota is adjusted by the disk space that is freed. The quota is also adjusted when `dm_write_invis` is invoked and additional disk space is consumed.

Since `dm_write_invis` runs with root credentials, it will never fail due to insufficient quota. However, it is possible that the quota of the file owner will be exceeded as a result of the invisible write. In that case the owner will not be able to perform further file operations that consume quota.

Memory mapped files

In GPFS, a read event or a write event will be generated (if enabled) at the time the memory mapping of a file is established.

No events will be generated during actual mapped access, regardless of the setting of the event list or the managed regions. Access to the file with regular file operations, while the file is memory mapped, will generate events, if such events are enabled.

To protect the integrity of memory mapped access, the DM application is not permitted to punch a hole in a file while the file is memory mapped. If the DM application calls `dm_punch_hole` while the file is memory mapped, the error code EBUSY will be returned.

Administration of IBM Spectrum Scale Data Management API for GPFS

To set up the DMAPI for GPFS, install the DMAPI files that are included in the GPFS installation package, and then choose the configuration attributes for DMAPI with the `mmchconfig` command. For each file system that you want DMAPI access, enable DMAPI with the `-z` flag of the `mmcrfs` or `mmchfs` command.

All DMAPI APIs must be called from nodes that are in the cluster where the file system is created. DMAPI APIs may `not` be invoked from a remote cluster. The GPFS daemon and each DMAPI application must be synchronized to prevent failures.

Administration of DMAPI for GPFS includes:

Required files for implementation of Data Management applications

The installation image for GPFS contains the required files for implementation of Data Management applications.

For more information about installation, see the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

The required files are:

dmapi.h
The header file that contains the C declarations of the DMAPI functions.

This header file must be included in the source files of the DM application.

The file is installed in directory: `/usr/lpp/mmfs/include`.

dmapi_types.h
The header file that contains the C declarations of the data types for the DMAPI functions and event messages.

The header file `dmapi.h` includes this header file.

The file is installed in directory: `/usr/lpp/mmfs/include`.

libdmapi.a
The library that contains the DMAPI functions.

The library `libdmapi.a` consists of a single shared object, which is built with auto-import of the system calls that are listed in the export file `dmapi.exp`.

The file is installed in directory: `/usr/lpp/mmfs/lib`.

dmapi.exp
The export file that contains the DMAPI system call names.

The file `dmapi.exp` needs to be explicitly used only if the DM application is to be explicitly built with static binding, using the binder options `-bnso -bI:dmapi.exp`.

The file is installed in directory: `/usr/lpp/mmfs/lib`.

dmapicalls, dmapicalls64
Module loaded during processing of the DMAPI functions.

The module is installed in directory: `/usr/lpp/mmfs/bin`.

Notes:

- On Linux nodes running DMAPI, the required files `libdmapi.a`, `dmapi.exp`, `dmapicalls`, and `dmapicalls64` are replaced by `libdmapi.so`.
- If you are compiling with a non-IBM compiler on AIX nodes, you must compile DMAPI applications with -D_AIX.

GPFS configuration attributes for DMAPI

GPFS uses several attributes for DMAPI that define various timeout intervals. These attributes can be changed with the `mmchconfig` command.

The DMAPI configuration attributes are:

dmapiDataEventRetry
Controls how GPFS handles the data event when it is enabled again right after this event is handled by the DMAPI application. Valid values are:

-1
Specifies that GPFS will always regenerate the event as long as it is enabled. This value should only be used when the DMAPI application recalls and migrates the same file in parallel by many processes at the same time.

0
Specifies to never regenerate the event. This value should not be used if a file could be migrated and recalled at the same time.

Positive Number
Specifies how many times the data event should be retried. The default is 2, which should be enough to cover most DMAPI applications. Unless a special situation occurs, you can increase this to a larger number or even set this to **-1** to always regenerate the events. Unless you perform careful testing, IBM recommends that you never change the default setting.

dmapiEventTimeout
Controls the blocking of file operation threads of NFS, while in the kernel waiting for the handling of a DMAPI synchronous event. The parameter value is the maximum time, in milliseconds, the thread will block. When this time expires, the file operation returns **ENOTREADY**, and the event continues asynchronously. The NFS server is expected to repeatedly retry the operation, which eventually will find the response of the original event and continue. This mechanism applies only to read, write, and truncate events, and only when such events come from NFS server threads.

The timeout value is given in milliseconds. The value 0 indicates immediate timeout (fully asynchronous event). A value greater than or equal to 86400000 (which is 24 hours) is considered 'infinity' (no timeout, fully synchronous event). The default value is 86400000. See also “Support for NFS” on page 812.

For the parameter change to take effect, restart the GPFS daemon on all nodes.

dmapiFileHandleSize
Controls the size of file handles generated by GPFS. The default DMAPI file handle size is 32 bytes. For clusters created prior to GPFS 3.2, the default DMAPI file handle size is 16 bytes.

Note: To change the DMAPI file handle size, GPFS must be stopped on all nodes in the cluster.

dmapiMountEvent
Controls the generation of the `mount`, `preunmount`, and `unmount` events. Valid values are:

all
Specifies that `mount`, `preunmount`, and `unmount` events are generated on each node. This is the default behavior.

LocalNode
Specifies that `mount`, `preunmount`, and `unmount` events are generated only if the node is a session node.

SessionNode
Specifies that `mount`, `preunmount`, and `unmount` events are generated on each node and are delivered to the session node, but the session node will respond with `DM_RESP_CONTINUE` to the event node without delivering the event to the DMAPI application, unless the event is originated from the `SessionNode` itself.

dmapiMountTimeout
Controls the blocking of mount operations, waiting for a disposition for the mount event to be set. This timeout is activated at most once on each node, by the first mount of a file system which has DMAPI enabled, and only if there has never before been a mount disposition. Any mount operation on this node that starts while the timeout period is active will wait for the mount disposition. The parameter value is the maximum time, in seconds, that the mount operation will wait for a disposition. When this time expires and there still is no disposition for the mount event, the mount operation fails, returning the **EIO** error.

The timeout value is given in full seconds. The value 0 indicates immediate timeout (immediate failure of the mount operation). A value greater than or equal to 86400 (which is 24 hours) is considered 'infinity' (no timeout, indefinite blocking until there is a disposition). The default value is 60. See also “Mount and unmount” on page 810 and “Initializing the Data Management application” on page 816.

dmapiSessionFailureTimeout
Controls the blocking of file operation threads, while in the kernel, waiting for the handling of a DMAPI synchronous event that is enqueued on a session that has suffered a failure. The parameter value is the maximum time, in seconds, the thread will wait for the recovery of the failed session. When this time expires and the session has not yet recovered, the event is aborted and the file operation fails, returning the **EIO** error.

The timeout value is given in full seconds. The value 0 indicates immediate timeout (immediate failure of the file operation). A value greater than or equal to 86400 (which is 24 hours) is considered 'infinity' (no timeout, indefinite blocking until the session recovers). The default value is 0. See also “Failure and recovery of IBM Spectrum Scale Data Management API for GPFS” on page 837 for details on session failure and recovery.

Enabling DMAPI for a file system

DMAPI must be enabled individually for each file system.

About this task

DMAPI can be enabled for a file system when the file system is created, using the `-z yes` option on the `mmcrfs` command. The default is `-z no`. The setting can be changed when the file system is not mounted anywhere, using the `-z yes | no` option on the `mmchfs` command. The setting is persistent.

The current setting can be queried using the `-z` option on the `mmlsfs` command.

While DMAPI is disabled for a given file system, no events are generated by file operations of that file system. Any DMAPI function calls referencing that file system fail with an EPERM error.

When **mmchfs -z no** is used to disable DMAPI, existing event lists, extended attributes, and managed regions in the given file system remain defined, but will be ignored until DMAPI is re-enabled. The command **mmchfs -z no** should be used with caution, since punched holes, if any, are no longer protected by managed regions.

For more information about GPFS commands, see Chapter 1, “Command reference,” on page 1.

Initializing the Data Management application

All DMAPI APIs must be called from nodes that are in the cluster where the file system is created. DMAPI APIs may `not` be invoked from a remote cluster.

About this task

During initialization of GPFS, it is necessary to synchronize the GPFS daemon and the DM application to prevent mount operations from failing. There are two mechanisms to accomplish this:

1. The shell script **gpfsready** invoked by the GPFS daemon during initialization.
2. A timeout interval, allowing mount operations to wait for a disposition to be set for the mount event.

During GPFS initialization, the daemon invokes the shell script **gpfsready**, located in directory `/var/mmfs/etc`. This occurs as the file systems are starting to be mounted. The shell script can be modified to start or restart the DM application. Upon return from this script, a session should have been created and a disposition set for the mount event. Otherwise, mount operations may fail due to a lack of disposition.

In a multiple-node environment such as GPFS, usually only a small subset of the nodes are session nodes, having DM applications running locally. On a node that is not a session node, the **gpfsready** script can be modified to synchronize between the local GPFS daemon and a remote DM application. This will prevent mount from failing on any node.

A sample shell script `gpfsready.sample` is installed in directory `/usr/lpp/mmfs/samples`.

If no mount disposition has ever been set in the cluster, the first external mount of a DMAPI-enabled file system on each node will activate a timeout interval on that node. Any mount operation on that node that starts during the timeout interval will wait for the mount disposition until the timeout expires. The timeout interval is configurable using the `dmapiMountTimeout` configuration attribute on the `mmchconfig` command (the interval can even be made infinite). A message is displayed at the beginning of the wait. If there is still no disposition for the mount event when the timeout expires, the mount operation will fail with an **EIO** error code. See “GPFS configuration attributes for DMAPI” on page 814 for more information on `dmapiMountTimeout`.

Specifications of enhancements for IBM Spectrum Scale Data Management API for GPFS

DMAPI for GPFS provides numerous enhancements in data structures and functions.

These enhancements are provided mainly by the multiple-node environment. Some data structures have additional fields. Many functions have usage restrictions, changes in semantics, and additional error codes. The enhancements are in these areas:

Enhancements to data structures

This is a description of GPFS enhancements to data structures defined in the XDSM standard.

For complete C declarations of all the data structures that are used in DMAPI for GPFS, refer to the `dmapi_types.h` file located in the **/usr/lpp/mmfs/include** directory.

- All file offsets and sizes in DMAPI data structures are 64 bits long.
- Names or path names that are passed in event messages are character strings, terminated by a null character. The length of the name buffer, as specified in the `dm_vardata_t` structure, includes the null character.
- The `dm_region_t` structure has a new 4-byte field, `rg_opaque`. The DMAPI implementation does not interpret `rg_opaque`. The DM application can use this field to store additional information within the managed region.
- The `dt_change` field in the `dm_stat` structure is not implemented in the inode. The value will change each time it is returned by the `dm_get_fileattr` function.
- The `dt_dtime` field in the `dm_stat` structure is overloaded on the `dt_ctime` field.
- The `dm_eventmsg` structure has a 4 byte field, `ev_nodeid` that uniquely identifies the node that generated the event. The id is the GPFS cluster data node number, which is attribute `node_number` in the `mmsdrfs2` file for a PSSP node or `mmsdrfs` file for any other type of node.
- The `ne_mode` field in the `dm_namesp_event` structure has an additional flag, `DM_LOCAL_MOUNT`. For the events preunmount and unmount when this flag is set, the unmount operation is local to the session node. See "Mount and unmount" on page 810. The `me_mode` field in the `dm_mount_event` structure has two additional flags; `DM_LOCAL_MOUNT`, and `DM_REMOTE_MOUNT`. See "Mount and unmount" on page 810.
- There are two 'quick access' single-bit opaque DM attributes for each file, stored directly in the inode. See "Data Management attributes" on page 812.
- The data type `dm_eventset_t` is implemented as a bit map, containing one bit for each event that is defined in DMAPI. The bit is set if, and only if, the event is present.

  Variables of type `dm_eventset_t` should be manipulated only using special macros. The XDSM standard provides a basic set of such macros. GPFS provides a number of additional macros. The names of all such macros begin with the prefix `DMEV_`.

  This is the list of additional macros that are provided in DMAPI for GPFS:

  DMEV_ALL(eset)
  Add all events to `eset`

  DMEV_ISZERO(eset)
  Check if `eset` is empty

DMEV_ISALL(eset)
Check if `eset` contains all events

DMEV_ADD(eset1, eset2)
Add to `eset2` all events in `eset1`

DMEV_REM(eset1, eset2)
Remove from `eset2` all events in `eset1`

DMEV_RES(eset1, eset2)
Restrict `eset2` by `eset1`

DMEV_ISEQ(eset1, eset2)
Check if `eset1` and `eset2` are equal

DMEV_ISDISJ(eset1, eset2)
Check if `eset1` and `eset2` are disjoint

DMEV_ISSUB(eset2)
Check if `eset1` is a subset of `eset2`

DMEV_NORM(eset)
Normalize the internal format of `eset`, clearing all unused bits

- DMAPI for GPFS provides a set of macros for comparison of token ids (value of type `dm_token_t`).

DM_TOKEN_EQ (x,y)
Check if **x** and **y** are the same

DM_TOKEN_NE (x,y)
Check if **x** and **y** are different

DM_TOKEN_LT (x,y)
Check if **x** is less than **y**

DM_TOKEN_GT (x,y)
Check if **x** is greater than **y**

DM_TOKEN_LE (x,y)
Check if **x** is less than or equal to **y**

DM_TOKEN_GE (x,y)
Check if **x** is greater than or equal to **y**

Usage restrictions on DMAPI functions

There are usage restrictions on the DMAPI for GPFS functions.

- The maximum number of DMAPI sessions that can be created on a node is 4000.
- Root credentials are a prerequisite for invoking any DMAPI function, otherwise the function fails with an **EPERM** error code.
- DMAPI functions are unable to run if the GPFS kernel extension is not loaded, or if the runtime module `dmapicalls` is not installed. An **ENOSYS** error code is returned in this case.
- Invoking a DMAPI function that is not implemented in GPFS results in returning the **ENOSYS** error code.
- DMAPI functions will fail, with the **ENOTREADY** error code, if the local GPFS daemon is not running.
- DMAPI functions will fail, with the **EPERM** error code, if DMAPI is disabled for the file system that is referenced by the file handle argument.
- DMAPI functions cannot access GPFS reserved files, such as quota files, inode allocation maps, and so forth. The **EBADF** error code is returned in this case.
- GPFS does not support access rights on entire file systems (as opposed to individual files). Hence, DMAPI function calls that reference a file system (with a file system handle) cannot present a token, and must use `DM_NO_TOKEN`. Functions affected by this restriction are:
  - `dm_set_eventlist`
  - `dm_get_eventlist`

  - dm_set_disp
  - dm_get_mountinfo
  - dm_set_return_on_destroy
  - dm_get_bulkattr
  - dm_get_bulkall

  If a token is presented, these functions fail with the **EINVAL** error code.
- DMAPI functions that acquire, change, query, or release access rights, must not present a file system handle. These functions are:
  - dm_request_right
  - dm_upgrade_right
  - dm_downgrade_right
  - dm_release_right
  - dm_query_right

  If a file system handle is presented, these functions fail with the **EINVAL** error code.
- The function dm_request_right, when invoked without wait (the *flags* argument has a value of 0), will almost always fail with the **EAGAIN** error. A GPFS implementation constraint prevents this function from completing successfully without wait, even if it is known that the requested access right is available. The DM_RR_WAIT flag must always be used. If the access right is available, there will be no noticeable delay.
- DMAPI function calls that reference a specific token, either as input or as output, can be made only on the session node. Otherwise, the call fails with the **EINVAL** error code.
- DMAPI function calls that reference an individual file by handle must be made on the session node. The corresponding file system must be mounted on the session node. The call fails with **EINVAL** if it is not on the session node, and with **EBADF** if the file system is not mounted.
- DMAPI function calls that reference a file system by handle (as opposed to an individual file) can be made on any node, not just the session node. The relevant functions are:
  - dm_set_eventlist
  - dm_get_eventlist
  - dm_set_disp
  - dm_get_mountinfo
  - dm_set_return_on_destroy
  - dm_get_bulkattr
  - dm_get_bulkall

  For dm_get_bulkattr and dm_get_bulkall, the system file must be mounted on the node that is making the call. For the other functions, the file system must be mounted on some node, but not necessarily on the node that is making the call. As specified previously, all such function calls must use DM_NO_TOKEN. The function fails with the **EBADF** error code if the file system is not mounted as required.
- The function dm_punch_hole will fail with the **EBUSY** error code if the file to be punched is currently memory-mapped.
- The function dm_move_event can only be used when the source session and the target session are on the same node. The function must be called on the session node. Otherwise, the function fails with the **EINVAL** error code.
- The function dm_create_session, when providing an existing session id in the argument *oldsid*, can only be called on the session node, except after session node failure. Otherwise, the call will return the **EINVAL** error code.

- The function `dm_destroy_session` can only be called on the session node, otherwise the call will fail with the **EINVAL** error code.
- The function `dm_set_fileattr` cannot change the file size. If the `dm_at_size` bit in the attribute mask is set, the call fails with the **EINVAL** error code.
- DMAPI functions that reference an event with a token fail with the **ESRCH** error code, if the event is not in an outstanding state. This is related to session recovery. See "Failure and recovery of IBM Spectrum Scale Data Management API for GPFS" on page 837 for details on session failure and recovery.

For additional information about:

- Semantic changes to the DMAPI for GPFS functions, see "Semantic changes to DMAPI functions" on page 833.
- C declarations of all functions in DMAPI for GPFS, refer to the `dmapi.h` file located in the **/usr/lpp/mmfs/include** directory.

Definitions for GPFS-specific DMAPI functions

The GPFS-specific DMAPI functions are not part of the DMAPI open standard.

You can use the following GPFS-specific DMAPI functions to work with file system snapshots:

- "dm_handle_to_snap" on page 821
- "dm_make_xhandle" on page 822

You can use the following GPFS-specific DMAPI functions to make asynchronous updates to attributes, managed regions, and event lists on files:

- "dm_remove_dmattr_nosync" on page 824
- "dm_set_dmattr_nosync" on page 826
- "dm_set_eventlist_nosync" on page 828
- "dm_set_region_nosync" on page 830

You can use the following GPFS-specific DMAPI function to make the previously listed asynchronous updates persistent by flushing them to disk:

- "dm_sync_dmattr_by_handle" on page 832

dm_handle_to_snap

Extracts a snapshot ID from a handle.

Synopsis

```
int dm_handle_to_snap(
    void          *hanp,          /* IN */
    size_t        hlen,           /* IN */
    dm_snap_t     *isnapp         /* OUT */
);
```

Description

Use the `dm_handle_to_snap` function to extract a snapshot ID from a handle. `dm_handle_to_snap()` is a GPFS-specific DMAPI function. It is not part of the open standard.

Parameters

void *_hanp_ (IN)
A pointer to an opaque DM handle previously returned by DMAPI.

size_t _hlen_ (IN)
The length of the handle in bytes.

dm_snap_t *_isnapp_ (OUT)
A pointer to the snapshot ID.

Return values

Zero is returned on success. On error, -1 is returned, and the global *errno* is set to one of the following values:

[EBADF]
The file handle does not refer to an existing or accessible object.

[EFAULT]
The system detected an invalid address in attempting to use an argument.

[EINVAL]
The argument *token* is not a valid token.

[ENOMEM]
DMAPI could not obtain the required resources to complete the call.

[ENOSYS]
Function is not supported by the DM implementation.

[EPERM]
The caller does not hold the appropriate privilege.

See also

“dm_make_xhandle” on page 822

dm_make_xhandle

Converts a file system ID, inode number, inode generation count, and snapshot ID into a handle.

Synopsis

```
int dm_make_xhandle(
    dm_fsid_t               *fsidp,         /* IN */
    dm_ino_t                *inop,          /* IN */
    dm_igen_t               *igenp,         /* IN */
    dm_snap_t               *isnapp,        /* IN */
    void                    **hanpp,        /* OUT */
    size_t                  *hlenp          /* OUT */
);
```

Description

Use the `dm_make_xhandle()` function to convert a file system ID, inode number, inode generation count, and snapshot ID into a handle. `dm_make_xhandle()` is a GPFS-specific DMAPI function. It is not part of the open standard.

Parameters

dm_fsid_t *_fsidp_ (IN)
The file system ID.

dm_ino_t *_inop_ (IN)
The inode number.

dm_igen_t *_igenp_ (IN)
The inode generation count.

dm_snap_t *_isnapp_ (IN)
The snapshot ID.

void **_hanpp_ (OUT)
A DMAPI initialized pointer that identifies a region of memory containing an opaque DM handle. The caller is responsible for freeing the allocated memory.

size_t *_hlenp_ (OUT)
The length of the handle in bytes.

Return values

Zero is returned on success. On error, -1 is returned, and the global *errno* is set to one of the following values:

[EBADF]
The file handle does not refer to an existing or accessible object.

[EFAULT]
The system detected an invalid address in attempting to use an argument.

[EINVAL]
The argument *token* is not a valid token.

[ENOMEM]
DMAPI could not obtain the required resources to complete the call.

[ENOSYS]
Function is not supported by the DM implementation.

[EPERM]
The caller does not hold the appropriate privilege.

See also

“dm_handle_to_snap” on page 821

dm_remove_dmattr_nosync

Asynchronously removes the specified attribute.

Synopsis

```
int dm_remove_dmattr_nosync(
    dm_sessid_t      sid,
    void            *hanp,
    size_t           hlen,
    dm_token_t       token,
    int              setdtime,
    dm_attrname_t   *attrnamep
);
```

Description

Use the dm_remove_dmattr_nosync function to asynchronously remove the attribute specified by *attrname*.

dm_remove_dmattr_nosync is a GPFS-specific DMAPI function; it is not part of the open standard. It has the same purpose, parameters, and return values as the standard DMAPI dm_remove_dmattr function, except that the update that it performs is not persistent until some other activity on that file (or on other files in the file system) happens to flush it to disk. To be certain that your update is made persistent, use one of the following functions:

- Standard DMAPI dm_sync_by_handle function, which flushes the file data and attributes
- GPFS-specific dm_sync_dmattr_by_handle function, which flushes only the attributes.

Parameters

dm_sessid_t *sid* (IN)
The identifier for the session of interest.

void **hanp* (IN)
The handle for the file for which the attributes should be removed.

size_t *hlen* (IN)
The length of the handle in bytes.

dm_token_t **token* (IN)
The token referencing the access right for the handle. The access right must be DM_RIGHT_EXCL, or the token DM_NO_TOKEN may be used and the interface acquires the appropriate rights.

int *setdtime* (IN)
If *setdtime* is non-zero, updates the file's attribute time stamp.

dm_attrname_t **attrnamep* (IN)
The attribute to be removed.

Return values

Zero is returned on success. On error, -1 is returned, and the global *errno* is set to one of the following values:

[EACCES]
The access right referenced by the token for the handle is not DM_RIGHT_EXCL.

[EBADF]
The file handle does not refer to an existing or accessible object.

[EFAULT]
The system detected an invalid address in attempting to use an argument.

[EINVAL]
The argument *token* is not a valid token.

[EINVAL]
The session is not valid.

[EIO]
I/O error resulted in failure of operation.

[ENOSYS]
The DMAPI implementation does not support this optional function.

[EPERM]
The caller does not hold the appropriate privilege.

[EROFS]
The operation is not allowed on a read-only file system.

See also

“dm_set_dmattr_nosync” on page 826, “dm_sync_dmattr_by_handle” on page 832

dm_set_dmattr_nosync

Asynchronously creates or replaces the value of the named attribute with the specified data.

Synopsis

```
int dm_set_dmattr_nosync(
    dm_sessid_t     sid,
    void           *hanp,
    size_t          hlen,
    dm_token_t      token,
    dm_attrname_t  *attrnamep,
    int             setdtime,
    size_t          buflen,
    void           *bufp
);
```

Description

Use the `dm_set_dmattr_nosync` function to asynchronously create or replace the value of the named attribute with the specified data.

`dm_set_dmattr_nosync` is a GPFS-specific DMAPI function; it is not part of the open standard. It has the same purpose, parameters, and return values as the standard DMAPI `dm_set_dmattr` function, except that the update that it performs is not persistent until some other activity on that file (or on other files in the file system) happens to flush it to disk. To be certain that your update is made persistent, use one of the following functions:

- Standard DMAPI `dm_sync_by_handle` function, which flushes the file data and attributes
- GPFS-specific `dm_sync_dmattr_by_handle` function, which flushes only the attributes.

Parameters

dm_sessid_t *sid* (IN)
The identifier for the session of interest.

void **hanp* (IN)
The handle for the file for which the attributes should be created or replaced.

size_t *hlen* (IN)
The length of the handle in bytes.

dm_token_t **token* (IN)
The token referencing the access right for the handle. The access right must be `DM_RIGHT_EXCL`, or the token `DM_NO_TOKEN` may be used and the interface acquires the appropriate rights.

dm_attrname_t **attrnamep* (IN)
The attribute to be created or replaced.

int *setdtime* (IN)
If *setdtime* is non-zero, updates the file's attribute time stamp.

size_t *buflen* (IN)
The size of the buffer in bytes.

void **bufp* (IN)
The buffer containing the attribute data.

Return values

Zero is returned on success. On error, -1 is returned, and the global *errno* is set to one of the following values:

[E2BIG]
The attribute value exceeds one of the implementation defined storage limits.

[E2BIG]
buflen is larger than the implementation defined limit. The limit can be determined by calling the `dm_get_config()` function.

[EACCES]
The access right referenced by the token for the handle is not `DM_RIGHT_EXCL`.

[EBADF]
The file handle does not refer to an existing or accessible object.

[EFAULT]
The system detected an invalid address in attempting to use an argument.

[EIO]
An attempt to write the new or updated attribute resulted in an I/O error.

[EINVAL]
The argument *token* is not a valid token.

[EINVAL]
The session is not valid.

[ENOMEM]
The DMAPI could not acquire the required resources to complete the call.

[ENOSPC]
An attempt to write the new or updated attribute resulted in an error due to no free space being available on the device.

[ENOSYS]
The DMAPI implementation does not support this optional function.

[EPERM]
The caller does not hold the appropriate privilege.

[EROFS]
The operation is not allowed on a read-only file system.

See also

“dm_remove_dmattr_nosync” on page 824, “dm_sync_dmattr_by_handle” on page 832

dm_set_eventlist_nosync

Asynchronously sets the list of events to be enabled for an object.

Synopsis

```
int dm_set_eventlist_nosync(
    dm_sessid_t      sid,
    void            *hanp,
    size_t           hlen,
    dm_token_t       token,
    dm_eventset_t   *eventsetp,
    u_int            maxevent
);
```

Description

Use the `dm_set_eventlist_nosync` function to asynchronously set the list of events to be enabled for an object.

`dm_set_eventlist_nosync` is a GPFS-specific DMAPI function; it is not part of the open standard. It has the same purpose, parameters, and return values as the standard DMAPI `dm_set_eventlist` function, except that the update that it performs is not persistent until some other activity on that file (or on other files in the file system) happens to flush it to disk. To be certain that your update is made persistent, use one of the following functions:

- Standard DMAPI `dm_sync_by_handle` function, which flushes the file data and attributes
- GPFS-specific `dm_sync_dmattr_by_handle` function, which flushes only the attributes.

Parameters

dm_sessid_t *sid* (IN)
The identifier for the session of interest.

void ****hanp* (IN)
The handle for the object. The handle can be either the system handle or a file handle.

size_t *hlen* (IN)
The length of the handle in bytes.

dm_token_t ****token* (IN)
The token referencing the access right for the handle. The access right must be `DM_RIGHT_EXCL`, or the token `DM_NO_TOKEN` may be used and the interface acquires the appropriate rights.

dm_eventset_t ****eventsetp* (IN)
The list of events to be enabled for the object.

u_int *maxevent* (IN)
The number of events to be checked for dispositions in the event set. The events from 0 to *maxevent*`-1` are examined.

Return values

Zero is returned on success. On error, -1 is returned, and the global *errno* is set to one of the following values:

[EACCES]
The access right referenced by the token for the handle is not `DM_RIGHT_EXCL`.

[EBADF]
The file handle does not refer to an existing or accessible object.

[EFAULT]
The system detected an invalid address in attempting to use an argument.

[EINVAL]
The argument *token* is not a valid token.

[EINVAL]
The session is not valid.

[EINVAL]
Tried to set event on a global handle.

[ENOMEM]
The DMAPI could not acquire the required resources to complete the call.

[ENXIO]
The implementation of the DMAPI does not support enabling event delivery on the specified handle.

[EPERM]
The caller does not hold the appropriate privilege.

[EROFS]
The operation is not allowed on a read-only file system.

See also

"dm_sync_dmattr_by_handle" on page 832

dm_set_region_nosync

Asynchronously replaces the set of managed regions for a file.

Synopsis

```
int dm_set_region_nosync(
    dm_sessid_t     sid,
    void           *hanp,
    size_t          hlen,
    dm_token_t      token,
    u_int           nelem,
    dm_region_t    *regbufp,
    dm_boolean_t   *exactflagp
);
```

Description

Use the `dm_set_region_nosync` function to asynchronously replace the set of managed regions for a file.

`dm_set_region_nosync` is a GPFS-specific DMAPI function; it is not part of the open standard. It has the same purpose, parameters, and return values as the standard DMAPI `dm_set_region` function, except that the update that it performs is not persistent until some other activity on that file (or on other files in the file system) happens to flush it to disk. To be certain that your update is made persistent, use one of the following functions:

- Standard DMAPI `dm_sync_by_handle` function, which flushes the file data and attributes
- GPFS-specific `dm_sync_dmattr_by_handle` function, which flushes only the attributes.

Parameters

dm_sessid_t *sid* (IN)
The identifier for the session of interest.

void **hanp* (IN)
The handle for the regular file to be affected.

size_t *hlen* (IN)
The length of the handle in bytes.

dm_token_t **token* (IN)
The token referencing the access right for the handle. The access right must be `DM_RIGHT_EXCL`, or the token `DM_NO_TOKEN` may be used and the interface acquires the appropriate rights.

u_int *nelem* (IN)
The number of input regions in *regbufp*. If *nelem* is 0, then all existing managed regions are cleared.

dm_region_t **regbufp* (IN)
A pointer to the structure defining the regions to be set. May be NULL if *nelem* is zero.

dm_boolean_t **exactflagp* (OUT)
If `DM_TRUE`, the file system did not alter the requested managed region set.

Valid values for the *rg_flags* field of the region structure are created by OR'ing together one or more of the following values:

DM_REGION_READ
Enable synchronous event for read operations that overlap this managed region.

DM_REGION_WRITE
Enable synchronous event for write operations that overlap this managed region.

DM_REGION_TRUNCATE
Enable synchronous event for truncate operations that overlap this managed region.

DM_REGION_NOEVENT
Do not generate any events for this managed region.

Return values

Zero is returned on success. On error, -1 is returned, and the global *errno* is set to one of the following values:

[E2BIG]
The number of regions specified by nelem exceeded the implementation capacity.

[EACCES]
The access right referenced by the token for the handle is not `DM_RIGHT_EXCL`.

[EBADF]
The file handle does not refer to an existing or accessible object.

[EFAULT]
The system detected an invalid address in attempting to use an argument.

[EINVAL]
The argument *token* is not a valid token.

[EINVAL]
The file handle does not refer to a regular file.

[EINVAL]
The regions passed in are not valid because they overlap or some other problem.

[EINVAL]
The session is not valid.

[EIO]
An I/O error resulted in failure of operation.

[ENOMEM]
The DMAPI could not acquire the required resources to complete the call.

[EPERM]
The caller does not hold the appropriate privilege.

[EROFS]
The operation is not allowed on a read-only file system.

See also

“dm_sync_dmattr_by_handle” on page 832

dm_sync_dmattr_by_handle

Synchronizes one or more files' in-memory attributes with those on the physical medium.

Synopsis

```
int m_sync_dmattr_by_handle(
    dm_sessid_t    sid,
    void          *hanp,
    size_t         hlen,
    dm_token_t     token
);
```

Description

Use the dm_sync_dmattr_by_handle function to synchronize one or more files' in-memory attributes with those on the physical medium.

dm_sync_dmattr_by_handle is a GPFS-specific DMAPI function; it is not part of the open standard. It has the same purpose, parameters, and return values as the standard DMAPI dm_sync_by_handle function, except that it flushes only the attributes, not the file data.

Like dm_sync_by_handle, dm_sync_dmattr_by_handle commits all previously unsynchronized updates for that node, not just the updates for one file. Therefore, if you update a list of files and call dm_sync_dmattr_by_handle on the last file, the attribute updates to all of the files in the list are made persistent.

Parameters

dm_sessid_t *sid* (IN)
The identifier for the session of interest.

void ****hanp* **(IN)
The handle for the file whose attributes are to be synchronized.

size_t *hlen* (IN)
The length of the handle in bytes.

dm_token_t ****token* **(IN)
The token referencing the access right for the handle. The access right must be DM_RIGHT_EXCL, or the token DM_NO_TOKEN may be used and the interface acquires the appropriate rights.

Return values

Zero is returned on success. On error, -1 is returned, and the global *errno* is set to one of the following values:

[EACCES]
The access right referenced by the token for the handle is not DM_RIGHT_EXCL.

[EBADF]
The file handle does not refer to an existing or accessible object.

[EFAULT]
The system detected an invalid address in attempting to use an argument.

[EINVAL]
The argument *token* is not a valid token.

[ENOMEM]
The DMAPI could not acquire the required resources to complete the call.

[ENOSYS]
The DMAPI implementation does not support this optional function.

[EPERM]
The caller does not hold the appropriate privilege.

See also

"dm_remove_dmattr_nosync" on page 824, "dm_set_dmattr_nosync" on page 826, "dm_set_eventlist_nosync" on page 828, and "dm_set_region_nosync" on page 830

Semantic changes to DMAPI functions

There are semantic changes to functions in DMAPI for GPFS. These changes are entailed mostly by the multiple-node environment.

For a list of additional error codes that are used in DMAPI for GPFS, see "Additional error codes returned by DMAPI functions" on page 835. For C declarations of all the DMAPI for GPFS functions, refer to the `dmapi.h` file located in the **/usr/lpp/mmfs/include** directory.

- The following DMAPI functions can be invoked on any node, not just the session node, as long as the session exists on some node in the GPFS cluster.
  - `dm_getall_disp`
  - `dm_query_session`
  - `dm_send_msg`
- DMAPI functions that reference a file system, as opposed to an individual file, can be made on any node, not just the session node. Being able to call certain functions on any node has advantages. The DM application can establish event monitoring when receiving a mount event from any node. Also, a distributed DM application can change event lists and dispositions of any file system from any node.
  - `dm_set_eventlist`
  - `dm_get_eventlist`
  - `dm_set_disp`
  - `dm_get_mount_info`
  - `dm_set_return_on_destroy`
  - `dm_get_bulkattr`
  - `dm_get_bulkall`
- The following functions, that construct a handle from its components, do not check if the resulting handle references a valid file. Validity is checked when the handle is presented in function calls that actually reference the file.
  - `dm_make_handle`
  - `dm_make_fshandle`
  - `dm_make_xhandle`
- The following data movement functions may be invoked on any node within the GPFS cluster, provided they are run as root and present a session ID for an established session on the session node. For guidelines on how to perform data movement from multiple nodes, see "Parallelism in Data Management applications" on page 811.
  - `dm_read_invis`
  - `dm_write_invis`
  - `dm_probe_hole`
  - `dm_punch_hole`
- The following functions that extract components of the handle, do not check whether the specified handle references a valid file. Validity is checked when the handle is presented in function calls that actually reference the file.
  - `dm_handle_to_fsid`

  - dm_handle_to_igen
  - dm_handle_to_ino
  - dm_handle_to_snap
- dm_handle_to_fshandle converts a file handle to a file system handle without checking the validity of either handle.
- dm_handle_is_valid does not check if the handle references a valid file. It verifies only that the internal format of the handle is correct.
- dm_init_attrloc ignores all of its arguments, except the output argument *locp*. In DMAPI for GPFS, the location pointer is initialized to a constant. Validation of the session, token, and handle arguments is done by the bulk access functions.
- When dm_query_session is called on a node other than the session node, it returns only the first eight bytes of the session information string.
- dm_create_session can be used to move an existing session to another node, if the current session node has failed. The call must be made on the new session node. See "Failure and recovery of IBM Spectrum Scale Data Management API for GPFS" on page 837 for details on session node failure and recovery.
- Assuming an existing session, using **dm_create_session** does not change the session id. If the argument *sessinfop* is NULL, the session information string is not changed.
- The argument *maxevent* in the functions dm_set_disp and dm_set_eventlist is ignored. In GPFS the set of events is implemented as a bitmap, containing a bit for each possible event.
- The value pointed to by the argument *nelemp*, on return from the functions dm_get_eventlist and dm_get_config_events, is always **DM_EVENT_MAX-1**. The argument *nelem* in these functions is ignored.
- The *dt_nevents* field in the dm_stat_t structure, which is returned by the dm_get_fileattr and dm_get_bulkall functions, has a value of **DM_EVENT_MAX-1** when the file has a file-system–wide event enabled by calling the dm_set_eventlist function. The value will always be **3** when there is no file-system–wide event enabled. A value of **3** indicates that there could be a managed region enabled for the specific file, which might have enabled a maximum of three events: READ, WRITE, and TRUNCATE.
- The functions dm_get_config and dm_get_config_events ignore the arguments *hanp* and *hlen*. This is because the configuration is not dependent on the specific file or file system.
- The function dm_set_disp, when called with the global handle, ignores any events in the event set being presented, except the mount event. When dm_set_disp is called with a file system handle, it ignores the mount event.
- The function dm_handle_hash, when called with an individual file handle, returns the inode number of the file. When dm_handle_hash is called with a file system handle, it returns the value 0.
- The function dm_get_mountinfo returns two additional flags in the me_mode field in the dm_mount_event structure. The flags are DM_MOUNT_LOCAL and DM_MOUNT_REMOTE. See "Mount and unmount" on page 810 for details.

GPFS-specific DMAPI events

The GPFS-specific events are not part of the DMAPI open standard. You can use these GPFS events to filter out events that are not critical to file management and to prevent system overloads from trivial information.

The DMAPI standard specifies that the system must generate ATTRIBUTE events each time the "changed time" (**ctime**) attribute for a file changes. For systems that write files in parallel, like GPFS, this generates ATTRIBUTE events from every node writing to the file. Consequently, it is easy for ATTRIBUTE events to overwhelm a data management server. However, the only **ctime** changes that are critical to GPFS are changes to either the permissions or ACLs of a file. In most cases, GPFS can ignore other **ctime** changes.

To distinguish file permission and ACL changes from other **ctime** updates, the following DMAPI metadata attribute events allow GPFS to filter **ctime** updates. Using these events, DM servers are able to track file

permission changes without overwhelming the system with irrelevant ATTRIBUTE events. However, these events are not part of the CAE Specification C429 open standard and they were implemented specifically for GPFS systems.

Metadata Events

DM_EVENT_PREPERMCHANGE
Pre-permission change event. Event is triggered before file permission change.

DM_EVENT_POSTPERMCHANGE
Post-permission change event. Event is triggered after file permission change.

Note: If you only want to track permission and ACL changes, turn off the DM_EVENT_ATTRIBUTE and turn on both the DM_EVENT_PREPERMCHANGE and DM_EVENT_POSTPERMCHANGE events.

Additional error codes returned by DMAPI functions

DMAPI for GPFS uses additional error codes, not specified in the XDSM standard, for most DMAPI functions.

For C declarations of all the DMAPI for GPFS functions, refer to the dmapi.h file located in the **/usr/lpp/mmfs/include** directory.

For all DMAPI functions, these error codes are used:

ENOSYS
The GPFS kernel extension is not loaded, or the runtime module dmapicalls is not installed.

ENOSYS
An attempt has been made to invoke a DMAPI function that is not implemented in GPFS.

ENOTREADY
The local GPFS daemon is not running or is initializing.

ENOMEM
DMAPI could not acquire the required resources to complete the call. **ENOMEM** is defined in the XDSM standard for some DMAPI functions, but not for all.

ESTALE
An error has occurred which does not fit any other error code specified for this function.

For DMAPI functions that provide a file handle as an input argument, these error codes are used:

EINVAL
The format of the file handle is not valid.

This error is returned without attempting to locate any object that is referenced by the handle. The EINVAL error code is to be distinguished from the EBADF error code, which, as specified in the XDSM standard, indicates that the object does not exist or is inaccessible. Thus, GPFS provides a refinement, distinguishing between format and access errors related to handles.

EPERM
DMAPI is disabled for the file system that is referenced by the file handle.

For DMAPI functions that provide a token as an input argument, these error codes are used:

ESRCH
The event referenced by the token is not in outstanding state.

This is to be distinguished from the EINVAL error code, which is returned when the token itself is not valid. ESRCH is defined in the XDSM standard for some DMAPI functions, but not for all relevant functions. In GPFS, the ESRCH error code occurs mostly after recovery from session failure. See "Event recovery" on page 838 for details.

For these specific DMAPI functions, the error code listed is used:

Table 34. Specific DMAPI functions and associated error codes.

Name of function	Error codes and descriptions
dm_downgrade_right() dm_upgrade_right()	EINVAL - The session or token is not valid.
dm_get_region()	EPERM - The caller does not hold the appropriate privilege.
dm_init_service()	EFAULT - The system detected an invalid address in attempting to use an argument.
dm_move_event() dm_respond_event()	EINVAL - The token is not valid.
dm_punch_hole()	EBUSY - The file is currently memory mapped.
dm_probe_hole() dm_punch_hole()	EINVAL - The argument *len* is too large, and will overflow if cast into offset_t. EINVAL - The argument *off* is negative.
dm_write_invis()	EINVAL - The argument *flags* is not valid.
dm_read_invis() dm_write_invis()	EINVAL - The argument *len* is too large, and will overflow if placed into the uio_resid field in the structure **uio**. EINVAL - The argument *off* is negative.
dm_sync_by_handle()	EROFS - The operation is not allowed on a read-only file system.
dm_find_eventmsg() dm_get_bulkall() dm_get_bulkattr() dm_get_dirattrs() dm_get_events() dm_get_mountinfo() dm_getall_disp() dm_getall_dmattr() dm_handle_to_path()	EINVAL - The argument *buflen* is too large; it must be smaller than INT_MAX.
dm_get_alloc_info() dm_getall_sessions() dm_getall_tokens()	EINVAL - The argument *nelem* is too large; DMAPI cannot acquire sufficient resources.

Failure and recovery of IBM Spectrum Scale Data Management API for GPFS

Failure and recovery of DMAPI applications in the multiple-node GPFS environment is different than in a single-node environment.

The failure model in XDSM is intended for a single-node environment. In this model, there are two types of failures:

DM application failure
The DM application has failed, but the file system works normally. Recovery entails restarting the DM application, which then continues handling events. Unless the DM application recovers, events may remain pending indefinitely.

Total system failure
The file system has failed. All non-persistent DMAPI resources are lost. The DM application itself may or may not have failed. Sessions are not persistent, so recovery of events is not necessary. The file system cleans its state when it is restarted. There is no involvement of the DM application in such cleanup.

The simplistic XDSM failure model is inadequate for GPFS. In a multiple-node environment, GPFS can fail on one node, but survive on other nodes. This type of failure is called *single-node failure* (*or partial system failure*). GPFS is built to survive and recover from single-node failures, without meaningfully affecting file access on surviving nodes.

Designers of Data Management applications for GPFS must comply with the enhanced DMAPI failure model, in order to support recoverability of GPFS. These areas are addressed:

- "Single-node failure" on page 837
- "Session failure and recovery" on page 838
- "Event recovery" on page 838
- "Loss of access rights" on page 839
- "DODeferred deletions" on page 839
- "DM application failure" on page 839

Single-node failure

In DMAPI for GPFS, single-node failure means that DMAPI resources are lost on the failing node, but not on any other node.

The most common single-node failure is when the local GPFS daemon fails. This renders any GPFS file system at that node inaccessible. Another possible single-node failure is file system forced unmount. When just an individual file system is forced unmounted on some node, its resources are lost, but the sessions on that node, if any, survive.

Single-node failure has a different effect when it occurs on a session node or on a source node:

session node failure
When the GPFS daemon fails, all session queues are lost, as well as all nonpersistent local file system resources, particularly DM access rights. The DM application may or may not have failed. The missing resources may in turn cause DMAPI function calls to fail with errors such as **ENOTREADY** or **ESRCH**.

Events generated at other source nodes remain pending despite any failure at the session node. Also, client threads remain blocked on such events.

source node failure
Events generated by that node are obsolete. If such events have already been enqueued at the session node, the DM application will process them, even though this may be redundant since no client is waiting for the response.

According to the XDSM standard, sessions are not persistent. This is inadequate for GPFS. Sessions must be persistent to the extent of enabling recovery from single-node failures. This is in compliance with a

basic GPFS premise that single-node failures do not affect file access on surviving nodes. Consequently, after session node failure, the session queue and the events on it must be reconstructed, possibly on another node.

Session recovery is triggered by the actions of the DM application. The scenario depends on whether or not the DM application itself has failed.

If the DM application has failed, it must be restarted, possibly on another node, and assume the old session by id. This will trigger reconstruction of the session queue and the events on it, using backup information replicated on surviving nodes. The DM application may then continue handling events. The session id is never changed when a session is assumed.

If the DM application itself survives, it will notice that the session has failed by getting certain error codes from DMAPI function calls (**ENOTREADY**, **ESRCH**). The application could then be moved to another node and recover the session queue and events on it. Alternatively, the application could wait for the GPFS daemon to recover. There is also a possibility that the daemon will recover before the DM application even notices the failure. In these cases, session reconstruction is triggered when the DM application invokes the first DMAPI function after daemon recovery.

Session failure and recovery

A session fails when the GPFS daemon of the session node fails.

Session failure results in the loss of all DM access rights associated with events on the queue, and all the tokens become invalid. After the session has recovered, any previously outstanding synchronous events return to the initial (non-outstanding) state, and must be received again.

Session failure may also result in partial loss of the session information string. In such case, GPFS will be able to restore only the first eight characters of the session string. It is suggested to not have the DM application be dependent on more than eight characters of the session string.

In extreme situations, failure may also result in the loss of event dispositions for some file system. This happens only if the GPFS daemon fails simultaneously on all nodes where the file system was mounted. When the file system is remounted, a mount event will be generated, at which point the dispositions could be reestablished by the DM application.

During session failure, events originating from surviving nodes remain pending, and client threads remain blocked on such events. It is therefore essential that the DM application assume the old session and continue processing the pending events. To prevent indefinite blocking of clients, a mechanism has been implemented whereby pending events will be aborted and corresponding file operations failed with the **EIO** error if the failed session is not recovered within a specified time-out interval. The interval is configurable using the `dmapiSessionFailureTimeout` attribute on the `mmchconfig` command. See “GPFS configuration attributes for DMAPI” on page 814. The default is immediate timeout.

GPFS keeps the state of a failed session for 24 hours, during which the session should be assumed. When this time has elapsed, and the session has not been assumed, the session is discarded. An attempt to assume a session after it has been discarded will fail.

Event recovery

Synchronous events are recoverable after session failure.

The state of synchronous events is maintained both at the source node and at the session node. When the old session is assumed, pending synchronous events are resubmitted by surviving source nodes.

All the events originating from the session node itself are lost during session failure, including user events generated by the DM application. All file operations on the session node fail with the ESTALE error code.

When a session fails, all of its tokens become obsolete. After recovery, the `dm_getall_tokens` function returns an empty list of tokens, and it is therefore impossible to identify events that were outstanding when the failure occurred. All recovered events return to the initial non-received state, and must be explicitly received again. The token id of a recovered event is the same as prior to the failure (except for the mount event).

If the token of a recovered event is presented in any DMAPI function before the event is explicitly received again, the call will fail with the **ESRCH** error code. The **ESRCH** error indicates that the event exists, but is not in the outstanding state. This is to be distinguished from the **EINVAL** error code, which indicates that the token id itself is not valid (there is no event).

The semantics of the **ESRCH** error code in GPFS are different from the XDSM standard. This is entailed by the enhanced failure model. The DM application may not notice that the GPFS daemon has failed and recovered, and may attempt to use a token it has received prior to the failure. For example, it may try to respond to the event. The **ESRCH** error code tells the DM application that it must receive the event again, before it can continue using the token. Any access rights associated with the token prior to the failure are lost. See "Loss of access rights" on page 839.

When a mount event is resubmitted to a session during session recovery, it will have a different token id than before the failure. This is an exception to the normal behavior, since all other recovered events have the same token id as before. The DM application thus cannot distinguish between recovered and new mount events. This should not be a problem, since the DM application must in any case be able to handle multiple mount events for the same file system.

Unmount events will not be resubmitted after session recovery. All such events are lost. This should not be a problem, since the event cannot affect the unmount operation, which has already been completed by the time the event was generated. In other words, despite being synchronous, semantically the unmount event resembles an asynchronous post event.

Loss of access rights

When the GPFS daemon fails on the session node, all file systems on the node are forced unmounted. As a result, all DM access rights associated with any local session are lost.

After daemon recovery, when the old sessions are assumed and the events are resubmitted, there is no way of identifying events that were already being handled prior to the failure (outstanding events), nor is there a guarantee that objects have not been accessed or modified after the access rights were lost. The DM application must be able to recover consistently without depending on persistent access rights. For example, it could keep its own state of events in progress, or process events idempotently.

Similarly, when a specific file system is forced unmounted at the session node, all DM access rights associated with the file system are lost, although the events themselves prevail on the session queue. After the file system is remounted, DMAPI calls using existing tokens may fail due to insufficient access rights. Also, there is no guarantee that objects have not been accessed or modified after the access rights were lost.

DODeferred deletions

The asynchronous recovery code supports deferred deletions if there are no external mounts at the time of recovery.

Once a node successfully generates a mount event for an external mount, the `sgmgr` node will start deferred deletions if it is needed. Any internal mounts would bypass deferred deletions if the file system is DMAPI enabled.

DM application failure

If only the DM application fails, the session itself remains active, events remain pending, and client threads remain blocked waiting for a response. New events will continue to arrive at the session queue.

Note: GPFS is unable to detect that the DM application has failed.

The failed DM application must be recovered on the same node, and continue handling the events. Since no DMAPI resources are lost in this case, there is little purpose in moving the DM application to another node. Assuming an existing session on another node is not permitted in GPFS, except after session node failure.

If the DM application fails simultaneously with the session node, the `gpfsready` shell script can be used to restart the DM application on the failed node. See "Initializing the Data Management application" on

page 816. In the case of simultaneous failures, the DM application can also be moved to another node and assume the failed session there. See “Single-node failure” on page 837.

Chapter 3. GPFS programming interfaces

A list of all the GPFS programming interfaces and a short description of each is presented in this topic.

The GPFS APIs are not supported on Windows.

Table 35 on page 841 summarizes the GPFS programming interfaces.

Table 35. GPFS programming interfaces

Interface	Purpose
"gpfs_acl_t structure" on page 845	Contains buffer mapping for the `gpfs_getacl()` and `gpfs_putacl()` subroutines.
"gpfs_clone_copy() subroutine" on page 846	Creates a file clone of a read-only clone parent file.
"gpfs_clone_snap() subroutine" on page 848	Creates a read-only clone parent from a source file.
"gpfs_clone_split() subroutine" on page 850	Splits a file clone from its clone parent.
"gpfs_clone_unsnap() subroutine" on page 852	Changes a clone parent with no file clones back to a regular file.
"gpfs_close_inodescan() subroutine" on page 854	Closes an inode scan.
"gpfs_cmp_fssnapid() subroutine" on page 855	Compares two file system snapshot IDs.
"gpfs_declone() subroutine" on page 857	Removes file clone references to clone parent blocks.
"gpfs_direntx_t structure" on page 859	Contains attributes of a GPFS directory entry.
"gpfs_direntx64_t structure" on page 860	Contains attributes of a GPFS directory entry.
"gpfs_fcntl() subroutine" on page 862	Performs operations on an open file.
"gpfs_fgetattrs() subroutine" on page 865	Retrieves all extended file attributes in opaque format.
"gpfs_fputattrs() subroutine" on page 867	Sets all the extended file attributes for a file.
"gpfs_fputattrswithpathname() subroutine" on page 869	Sets all of the extended file attributes for a file and invokes the policy engine for RESTORE rules.
"gpfs_free_fssnaphandle() subroutine" on page 871	Frees a GPFS file system snapshot handle.
"gpfs_fssnap_handle_t structure" on page 872	Contains a handle for a GPFS file system or snapshot.
"gpfs_fssnap_id_t structure" on page 873	Contains a permanent identifier for a GPFS file system or snapshot.
"gpfs_fstat() subroutine" on page 874	Returns exact file status for a GPFS file.
"gpfs_fstat_x() subroutine" on page 876	Returns extended status information for a GPFS file with specified accuracy.
"gpfs_get_fsname_from_fssnaphandle() subroutine" on page 878	Obtains a file system name from its snapshot handle.
"gpfs_get_fssnaphandle_by_fssnapid() subroutine" on page 879	Obtains a file system snapshot handle using its snapshot ID.

Table 35. GPFS programming interfaces (continued)

Interface	Purpose
"gpfs_get_fssnaphandle_by_name() subroutine" on page 880	Obtains a file system snapshot handle using its name.
"gpfs_get_fssnaphandle_by_path() subroutine" on page 882	Obtains a file system snapshot handle using its path name.
"gpfs_get_fssnapid_from_fssnaphandle() subroutine" on page 884	Obtains a file system snapshot ID using its handle.
"gpfs_get_pathname_from_fssnaphandle() subroutine" on page 886	Obtains a file system path name using its snapshot handle.
"gpfs_get_snapdirname() subroutine" on page 888	Obtains the name of the directory containing global snapshots.
"gpfs_get_snapname_from_fssnaphandle() subroutine" on page 890	Obtains a snapshot name using its file system snapshot handle.
"gpfs_getacl() subroutine" on page 892	Retrieves the access control information for a GPFS file.
"gpfs_getacl_fd() subroutine" on page 894	Retrieves the access control information for a GPFS file.
"gpfs_iattr_t structure" on page 896	Contains attributes of a GPFS inode.
"gpfs_iattr64_t structure" on page 899	Contains attributes of a GPFS inode.
"gpfs_iclose() subroutine" on page 903	Closes a file given its inode file handle.
"gpfs_ifile_t structure" on page 904	Contains a handle for a GPFS inode.
"gpfs_igetattrs() subroutine" on page 905	Retrieves extended file attributes in opaque format.
"gpfs_igetattrsx() subroutine" on page 907	Retrieves extended file attributes; provides an option to include DMAPI attributes.
"gpfs_igetfilesetname() subroutine" on page 909	Returns the name of the fileset defined by a fileset ID.
"gpfs_igetstoragepool() subroutine" on page 911	Returns the name of the storage pool for the given storage pool ID.
"gpfs_iopen() subroutine" on page 913	Opens a file or directory by inode number.
"gpfs_iopen64() subroutine" on page 915	Opens a file or directory by inode number.
"gpfs_iputattrsx() subroutine" on page 917	Sets the extended file attributes for a file.
"gpfs_iread() subroutine" on page 920	Reads a file opened by `gpfs_iopen()`.
"gpfs_ireaddir() subroutine" on page 922	Reads the next directory entry.
"gpfs_ireaddir64() subroutine" on page 924	Reads the next directory entry.
"gpfs_ireadlink() subroutine" on page 926	Reads a symbolic link by inode number.
"gpfs_ireadlink64() subroutine" on page 928	Reads a symbolic link by inode number.
"gpfs_ireadx() subroutine" on page 930	Performs block level incremental read of a file within an incremental inode scan.
"gpfs_iscan_t structure" on page 932	Contains a handle for an inode scan of a GPFS file system or snapshot.
"gpfs_lib_init() subroutine" on page 933	Sets up a GPFS interface for additional calls.

Table 35. GPFS programming interfaces (continued)	
Interface	**Purpose**
“gpfs_lib_term() subroutine” on page 934	Cleans up after GPFS interface calls have been completed.
“gpfs_next_inode() subroutine” on page 935	Retrieves the next inode from the inode scan.
“gpfs_next_inode64() subroutine” on page 937	Retrieves the next inode from the inode scan.
“gpfs_next_inode_with_xattrs() subroutine” on page 939	Retrieves the next inode and its extended attributes from the inode scan.
“gpfs_next_inode_with_xattrs64() subroutine” on page 941	Retrieves the next inode and its extended attributes from the inode scan.
“gpfs_next_xattr() subroutine” on page 943	Returns individual attributes and their values.
“gpfs_opaque_acl_t structure” on page 945	Contains buffer mapping for the `gpfs_getacl()` and `gpfs_putacl()` subroutines.
“gpfs_open_inodescan() subroutine” on page 946	Opens an inode scan of a file system or snapshot.
“gpfs_open_inodescan64() subroutine” on page 949	Opens an inode scan of a file system or snapshot.
“gpfs_open_inodescan_with_xattrs() subroutine” on page 952	Opens an inode file and extended attributes for an inode scan.
“gpfs_open_inodescan_with_xattrs64() subroutine” on page 955	Opens an inode file and extended attributes for an inode scan.
“gpfs_prealloc() subroutine” on page 958	Preallocates storage for a regular file or preallocates directory entries for a directory.
“gpfs_putacl() subroutine” on page 961	Restores the access control information for a GPFS file.
“gpfs_putacl_fd() subroutine” on page 963	Restores the access control information for a GPFS file
“gpfs_quotactl() subroutine” on page 965	Manipulates disk quotas on file systems.
“gpfs_quotaInfo_t structure” on page 968	Contains buffer mapping for the `gpfs_quotactl()` subroutine.
“gpfs_seek_inode() subroutine” on page 970	Advances an inode scan to the specified inode number.
“gpfs_seek_inode64() subroutine” on page 972	Advances an inode scan to the specified inode number.
“gpfs_stat() subroutine” on page 974	Returns exact file status for a GPFS file.
“gpfs_stat_inode() subroutine” on page 976	Seeks the specified inode and retrieves that inode and its extended attributes from the inode scan.
“gpfs_stat_inode64() subroutine” on page 978	Seeks the specified inode and retrieves that inode and its extended attributes from the inode scan.
“gpfs_stat_inode_with_xattrs() subroutine” on page 980	Seeks the specified inode and retrieves that inode and its extended attributes from the inode scan.
“gpfs_stat_inode_with_xattrs64() subroutine” on page 982	Seeks the specified inode and retrieves that inode and its extended attributes from the inode scan.

Table 35. GPFS programming interfaces (continued)

Interface	Purpose
"gpfs_stat_x() subroutine" on page 984	Returns extended status information for a GPFS file with specified accuracy.
"gpfsFcntlHeader_t structure" on page 986	Contains declaration information for the `gpfs_fcntl()` subroutine.
"gpfsGetDataBlkDiskIdx_t structure" on page 987	Obtains the FPO data block location of a file.
"gpfsGetFilesetName_t structure" on page 989	Obtains the fileset name of a file.
"gpfsGetReplication_t structure" on page 990	Obtains the replication factors of a file.
"gpfsGetSetXAttr_t structure" on page 992	Obtains or sets extended attribute values.
"gpfsGetSnapshotName_t structure" on page 994	Obtains the snapshot name of a file.
"gpfsGetStoragePool_t structure" on page 995	Obtains the storage pool name of a file.
"gpfsListXAttr_t structure" on page 996	Lists extended attributes.
"gpfsRestripeData_t structure" on page 997	Restripes the data blocks of a file.
"gpfsSetReplication_t structure" on page 1004	Sets the replication factors of a file.
"gpfsSetStoragePool_t structure" on page 1006	Sets the assigned storage pool of a file.

gpfs_acl_t structure

Contains buffer mapping for the gpfs_getacl() and gpfs_putacl() subroutines.

Library

GPFS Library (libgpfs.a for AIX, libgpfs.so for Linux)

Structure

```
typedef struct gpfs_acl
{
  gpfs_aclLen_t     acl_len;     /* Total length of this ACL in bytes */
  gpfs_aclLevel_t   acl_level;   /* Reserved (must be zero) */
  gpfs_aclVersion_t acl_version; /* POSIX or NFS4 ACL */
  gpfs_aclType_t    acl_type;    /* Access, Default, or NFS4 */
  gpfs_aclCount_t   acl_nace;    /* Number of Entries that follow */
  union
  {
    gpfs_ace_v1_t  ace_v1[1]; /* when GPFS_ACL_VERSION_POSIX */
    gpfs_ace_v4_t  ace_v4[1]; /* when GPFS_ACL_VERSION_NFS4 */
    v4Level1_t     v4Level1;  /* when GPFS_ACL_LEVEL_V4FLAGS */
  };
} gpfs_acl_t;
```

Description

The gpfs_acl_t structure contains size, version, and ACL type information for the gpfs_getacl() and gpfs_putacl() subroutines.

Members

acl_len
The total length (in bytes) of this gpfs_acl_t structure.

acl_level
Reserved for future use. Currently must be zero.

acl_version
This field contains the version of the GPFS ACL. GPFS supports the following ACL versions: GPFS_ACL_VERSION_POSIX and GPFS_ACL_VERSION_NFS4. On input to the gpfs_getacl() subroutine, set this field to zero.

acl_type
On input to the gpfs_getacl() subroutine, set this field to:

- Either GPFS_ACL_TYPE_ACCESS or GPFS_ACL_TYPE_DEFAULT for POSIX ACLs
- GPFS_ACL_TYPE_NFS4 for NFS ACLs.

These constants are defined in the gpfs.h header file.

acl_nace
The number of ACL entries that are in the array (ace_v1 or ace_v4).

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX

/usr/lpp/mmfs/lib/libgpfs.so for Linux

gpfs_clone_copy() subroutine

Creates a file clone of a read-only clone parent file.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_clone_copy(const char *sourcePathP, const char *destPathP);
```

Description

The `gpfs_clone_copy()` subroutine creates a writeable file clone from a read-only clone parent file.

Parameters

sourcePathP
The path of a read-only source file to clone. The source file can be a file in a snapshot or a clone parent file created with the `gpfs_clone_snap()` subroutine.

destPathP
The path of the destination file to create. The destination file will become the file clone.

Exit status

If the `gpfs_clone_copy()` subroutine is successful, it returns a value of 0 and creates a file clone from the clone parent.

If the `gpfs_clone_copy()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EACCESS
Permission denied when writing to the destination path or reading from the source path.

EEXIST
The destination file already exists.

EFAULT
The input argument points outside the accessible address space.

EINVAL
The source or destination does not refer to a regular file or a GPFS file system.

EISDIR
The specified destination file is a directory.

ENAMETOOLONG
The source or destination path name is too long.

ENOENT
The source file does not exist.

ENOSPC
The file system has run out of disk space.

ENOSYS
The `gpfs_clone_copy()` subroutine is not available.

EPERM
The source file is a directory or is not a regular file.

EXDEV
The source file and destination file are not in the same file system.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

Related reference

“gpfs_clone_snap() subroutine” on page 848
Creates a read-only clone parent from a source file.

“gpfs_clone_split() subroutine” on page 850
Splits a file clone from its clone parent.

“gpfs_clone_unsnap() subroutine” on page 852
Changes a clone parent with no file clones back to a regular file.

gpfs_clone_snap() subroutine

Creates a read-only clone parent from a source file.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_clone_snap(const char *sourcePathP, const char *destPathP);
```

Description

The `gpfs_clone_snap()` subroutine creates a read-only clone parent from a source file.

Parameters

sourcePathP
The path of the source file to clone.

destPathP
The path of the destination file to create. The destination file will become a read-only clone parent file.

If `destPathP` is NULL, then the source file will be changed in place into a read-only clone parent. When using this method to create a clone parent, the specified file cannot be open for writing or have hard links.

Exit status

If the `gpfs_clone_snap()` subroutine is successful, it returns a value of 0 and creates a read-only clone parent from the source file.

If the `gpfs_clone_snap()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EACCESS
Permission denied when writing to the destination path or reading from the source path.

EEXIST
The destination file already exists.

EFAULT
The input argument points outside accessible address space.

EINVAL
The source or destination does not refer to a regular file or a GPFS file system.

EISDIR
The specified destination file is a directory.

ENAMETOOLONG
The source or destination path name is too long.

ENOENT
The source file does not exist.

ENOSPC
The file system has run out of disk space.

ENOSYS
The `gpfs_clone_snap()` subroutine is not available.

EPERM
The source file is a directory or is not a regular file, or you tried to create a clone file with depth greater than 1000.

EXDEV
The source file and destination file are not in the same file system.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

Related reference

"gpfs_clone_copy() subroutine" on page 846
Creates a file clone of a read-only clone parent file.

"gpfs_clone_split() subroutine" on page 850
Splits a file clone from its clone parent.

"gpfs_clone_unsnap() subroutine" on page 852
Changes a clone parent with no file clones back to a regular file.

gpfs_clone_split() subroutine

Splits a file clone from its clone parent.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_clone_split(gpfs_file_t fileDesc, int ancLimit);
```

Description

The `gpfs_clone_split()` subroutine splits a file clone from its clone parent. The `gpfs_declone()` subroutine must be called first to remove all references to the clone parent.

Parameters

fileDesc
File descriptor for the file clone to split from its clone parent.

ancLimit
The ancestor limit specified with one of these values:

GPFS_CLONE_ALL
Remove references to all clone parents.

GPFS_CLONE_PARENT_ONLY
Remove references from the immediate clone parent only.

Exit status

If the `gpfs_clone_split()` subroutine is successful, it returns a value of 0.

If the `gpfs_clone_split()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EACCESS
Permission denied when writing to the target file.

EBADF
The file descriptor is not valid or is not a GPFS file.

EINVAL
An argument to the function was not valid.

ENOSYS
The `gpfs_clone_split()` subroutine is not available.

EPERM
The file descriptor does not refer to a regular file or a file clone.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

Related reference

“gpfs_clone_copy() subroutine” on page 846
Creates a file clone of a read-only clone parent file.

“gpfs_clone_snap() subroutine” on page 848
Creates a read-only clone parent from a source file.

“gpfs_clone_unsnap() subroutine” on page 852
Changes a clone parent with no file clones back to a regular file.

gpfs_clone_unsnap() subroutine

Changes a clone parent with no file clones back to a regular file.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_clone_unsnap(gpfs_file_t fileDesc);
```

Description

The `gpfs_clone_unsnap()` subroutine changes a clone parent with no file clones back to a regular file.

Parameters

fileDesc
File descriptor for the clone parent to convert back to a regular file.

Exit status

If the `gpfs_clone_unsnap()` subroutine is successful, it returns a value of 0.

If the `gpfs_clone_unsnap()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EACCESS
Permission denied when writing to the target file.

EBADF
The file descriptor is not valid or is not a GPFS file.

EINVAL
An argument to the function was not valid.

ENOSYS
The `gpfs_clone_unsnap()` subroutine is not available.

EPERM
The file descriptor does not refer to a regular file or a clone parent.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

Related reference
“gpfs_clone_copy() subroutine” on page 846

Creates a file clone of a read-only clone parent file.

"gpfs_clone_snap() subroutine" on page 848
Creates a read-only clone parent from a source file.

"gpfs_clone_split() subroutine" on page 850
Splits a file clone from its clone parent.

gpfs_close_inodescan() subroutine

Closes an inode scan.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
void gpfs_close_inodescan(gpfs_iscan_t *iscan);
```

Description

The `gpfs_close_inodescan()` subroutine closes the scan of the inodes in a file system or snapshot that was opened with the `gpfs_open_inodescan()` subroutine. The `gpfs_close_inodescan()` subroutine frees all storage used for the inode scan and invalidates the `iscan` handle.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

iscan
Pointer to the inode scan handle.

Exit status

The `gpfs_close_inodescan()` subroutine returns void.

Exceptions

None.

Error status

None.

Examples

For an example using `gpfs_close_inodescan()`, see `/usr/lpp/mmfs/samples/util/tsgetusage.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_cmp_fssnapid() subroutine

Compares two file system snapshot IDs.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_cmp_fssnapid(const gpfs_fssnap_id_t *fssnapId1,
                      const gpfs_fssnap_id_t *fssnapId2,
                      int *result);
```

Description

The `gpfs_cmp_fssnapid()` subroutine compares two snapshot IDs for the same file system to determine the order in which the two snapshots were taken. The `result` parameter is set as follows:

- `result` less than zero indicates that snapshot 1 was taken before snapshot 2.
- `result` equal to zero indicates that snapshot 1 and 2 are the same.
- `result` greater than zero indicates that snapshot 1 was taken after snapshot 2.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fssnapId1
File system snapshot ID of the first snapshot.

fssnapId2
File system snapshot ID of the second snapshot.

result
Pointer to an integer indicating the outcome of the comparison.

Exit status

If the `gpfs_cmp_fssnapid()` subroutine is successful, it returns a value of 0 and the `result` parameter is set.

If the `gpfs_cmp_fssnapid()` subroutine is unsuccessful, it returns a value of -1 and the global error variable `errno` is set to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EDOM
The two snapshots cannot be compared because they were taken from two different file systems.

ENOSYS
The `gpfs_cmp_fssnapid()` subroutine is not available.

GPFS_E_INVAL_FSSNAPID
`fssnapId1` or `fssnapId2` is not a valid snapshot ID.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_declone() subroutine

Removes file clone references to clone parent blocks.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_declone(gpfs_file_t fileDesc, int ancLimit, gpfs_off64_t nBlocks,
                 gpfs_off64_t *offsetP);
```

Description

The `gpfs_declone()` subroutine removes all file clone references to a clone parent by copying the clone parent blocks to the file clone.

Parameters

fileDesc
The file descriptor for the file clone.

ancLimit
The ancestor limit specified with one of these values:

GPFS_CLONE_ALL
Remove references to all clone parents.

GPFS_CLONE_PARENT_ONLY
Remove references from the immediate clone parent only.

nBlocks
The maximum number of GPFS blocks to copy.

offsetP
A pointer to the starting offset within the file clone. This pointer will be updated to the offset of the next block to process or -1 if there no more blocks.

Exit status

If the `gpfs_declone()` subroutine is successful, it returns a value of 0.

If the `gpfs_declone()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EACCESS
Permission denied when writing to the target file.

EBADF
The file descriptor is not valid or is not a GPFS file.

EFAULT
The input argument points outside the accessible address space.

EINVAL
An argument to the function was not valid.

ENOSPC
The file system has run out of disk space.

ENOSYS
The `gpfs_declone()` subroutine is not available.

EPERM
The file descriptor does not refer to a regular file.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_direntx_t structure

Contains attributes of a GPFS directory entry.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Structure

```
typedef struct gpfs_direntx
{
  int            d_version;     /* this struct's version */
  unsigned short d_reclen;      /* actual size of this struct including
                                   null terminated variable length d_name */
  unsigned short d_type;        /* Types are defined below */
  gpfs_ino_t     d_ino;         /* File inode number */
  gpfs_gen_t     d_gen;         /* Generation number for the inode */
  char           d_name[256];   /* null terminated variable length name */
} gpfs_direntx_t;

/* File types for d_type field in gpfs_direntx_t */
#define GPFS_DE_OTHER    0
#define GPFS_DE_DIR      4
#define GPFS_DE_REG      8
#define GPFS_DE_LNK     10
#define GPFS_DE_DEL     16
```

Description

The `gpfs_direntx_t` structure contains the attributes of a GPFS directory entry.

Members

d_version
The version number of this structure.

d_reclen
The actual size of this structure including the null-terminated variable-length `d_name` field.

To allow some degree of forward compatibility, careful callers should use the `d_reclen` field for the size of the structure rather than the `sizeof()` function.

d_type
The type of directory.

d_ino
The directory inode number.

d_gen
The directory generation number.

d_name
Null-terminated variable-length name of the directory.

Examples

For an example using `gpfs_direntx_t`, see `/usr/lpp/mmfs/samples/util/tsfindinode.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_direntx64_t structure

Contains attributes of a GPFS directory entry.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Structure

```
typedef struct gpfs_direntx64
{
  int            d_version;     /* this struct's version */
  unsigned short d_reclen;      /* actual size of this struct including
                                   null terminated variable length d_name */
  unsigned short d_type;        /* Types are defined below */
  gpfs_ino64_t   d_ino;         /* File inode number */
  gpfs_gen64_t   d_gen;         /* Generation number for the inode */
  unsigned int   d_flags;       /* Flags are defined below */
  char           d_name[1028];  /* null terminated variable length name */
                                /* (1020+null+7 byte pad to double word) */
                                /* to handle up to 255 UTF-8 chars */
} gpfs_direntx64_t;

/* File types for d_type field in gpfs_direntx64_t */
#define GPFS_DE_OTHER    0
#define GPFS_DE_DIR      4
#define GPFS_DE_REG      8
#define GPFS_DE_LNK     10
#define GPFS_DE_DEL     16

/* Define flags for gpfs_direntx64_t */
#define GPFS_DEFLAG_NONE      0x0000 /* Default value, no flags set */
#define GPFS_DEFLAG_JUNCTION  0x0001 /* DirEnt is a fileset junction */
#define GPFS_DEFLAG_IJUNCTION 0x0002 /* DirEnt is a inode space junction */
#define GPFS_DEFLAG_ORPHAN    0x0004 /* DirEnt is an orphan (pcache) */
```

Description

The `gpfs_direntx64_t` structure contains the attributes of a GPFS directory entry.

Members

d_version
The version number of this structure.

d_reclen
The actual size of this structure including the null-terminated variable-length `d_name` field.

To allow some degree of forward compatibility, careful callers should use the `d_reclen` field for the size of the structure rather than the `sizeof()` function.

d_type
The type of directory.

d_ino
The directory inode number.

d_gen
The directory generation number.

d_flags
The directory flags.

d_name
Null-terminated variable-length name of the directory.

Examples

See the `gpfs_direntx_t` example in `/usr/lpp/mmfs/samples/util/tsfindinode.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_fcntl() subroutine

Performs operations on an open file.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_fcntl(gpfs_file_t fileDesc, void* fcntlArgP);
```

Description

The `gpfs_fcntl()` subroutine is used to pass file access pattern information and to control certain file attributes on behalf of an open file. More than one operation can be requested with a single invocation of `gpfs_fcntl()`. The type and number of operations is determined by the second operand, `fcntlArgP`, which is a pointer to a data structure built in memory by the application. This data structure consists of:

- A fixed length header, mapped by `gpfsFcntlHeader_t`.
- A variable list of individual file access hints, directives or other control structures:
  - File access hints:
    - `gpfsAccessRange_t`
    - `gpfsClearFileCache_t`
    - `gpfsFreeRange_t`
    - `gpfsMultipleAccessRange_t`
  - File access directives:
    - `gpfsCancelHints_t`
  - Platform-independent extended attribute operations:
    - `gpfsGetSetXAttr_t`
    - `gpfsListXAttr_t`
  - Other file attribute operations:
    - `gpfsGetDataBlkDiskIdx_t`
    - `gpfsGetFilesetName_t`
    - `gpfsGetReplication_t`
    - `gpfsGetSnapshotName_t`
    - `gpfsGetStoragePool_t`
    - `gpfsRestripeData_t`
    - `gpfsSetReplication_t`
    - `gpfsSetStoragePool_t`

  These hints, directives and other operations may be mixed within a single `gpfs_fcntl()` subroutine, and are performed in the order that they appear. A subsequent hint or directive may cancel out a preceding one.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fileDesc
The file descriptor identifying the file to which GPFS applies the hints and directives.

fcntlArgP
A pointer to the list of operations to be passed to GPFS.

Exit status

If the `gpfs_fcntl()` subroutine is successful, it returns a value of 0.

If the `gpfs_fcntl()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EBADF
The file descriptor is not valid.

EINVAL
The file descriptor does not refer to a GPFS file or a regular file.

The system call is not valid.

ENOSYS
The `gpfs_fcntl()` subroutine is not supported under the current file system format.

Examples

1. This programming segment releases all cache data held by the file *handle* and tell GPFS that the subroutine will write the portion of the file with file offsets between 2 GB and 3 GB minus one:

```
struct
{
  gpfsFcntlHeader_t hdr;
  gpfsClearFileCache_t rel;
  gpfsAccessRange_t acc;
} arg;

arg.hdr.totalLength = sizeof(arg);
arg.hdr.fcntlVersion = GPFS_FCNTL_CURRENT_VERSION;
arg.hdr.fcntlReserved = 0;
arg.rel.structLen = sizeof(arg.rel);
arg.rel.structType = GPFS_CLEAR_FILE_CACHE;
arg.acc.structLen = sizeof(arg.acc);
arg.acc.structType = GPFS_ACCESS_RANGE;
arg.acc.start = 2LL * 1024LL * 1024LL * 1024LL;
arg.acc.length = 1024 * 1024 * 1024;
arg.acc.isWrite = 1;
rc = gpfs_fcntl(handle, &arg);
```

2. This programming segment gets the storage pool name and fileset name of a file from GPFS.

```
struct {
gpfsFcntlHeader_t hdr;
gpfsGetStoragePool_t pool;
gpfsGetFilesetName_t fileset;
} fcntlArg;
fcntlArg.hdr.totalLength = sizeof(fcntlArg.hdr) + sizeof(fcntlArg.pool) +
sizeof(fcntlArg.fileset);
```

```
fcntlArg.hdr.fcntlVersion = GPFS_FCNTL_CURRENT_VERSION;
fcntlArg.hdr.fcntlReserved = 0;

fcntlArg.pool.structLen = sizeof(fcntlArg.pool);
fcntlArg.pool.structType = GPFS_FCNTL_GET_STORAGEPOOL;

fcntlArg.fileset.structLen = sizeof(fcntlArg.fileset);
fcntlArg.fileset.structType = GPFS_FCNTL_GET_FILESETNAME;

rc = gpfs_fcntl(fd, &fcntlArg);
```

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX

/usr/lpp/mmfs/lib/libgpfs.so for Linux

gpfs_fgetattrs() subroutine

Retrieves all extended file attributes in opaque format.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_fgetattrs(gpfs_file_t fileDesc,
                   int flags,
                   void *bufferP,
                   int bufferSize,
                   int *attrSizeP);
```

Description

The `gpfs_fgetattrs()` subroutine, together with `gpfs_fputattrs()`, is intended for use by a backup program to save (`gpfs_fgetattrs()`) and restore (`gpfs_fputattrs()`) extended file attributes such as ACLs, DMAPI attributes, and other information for the file. If the file has no extended attributes, the `gpfs_fgetattrs()` subroutine returns a value of 0, but sets `attrSizeP` to 0 and leaves the contents of the buffer unchanged.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fileDesc
The file descriptor identifying the file whose extended attributes are being retrieved.

flags
Must have one of the following values:

GPFS_ATTRFLAG_DEFAULT
Saves the attributes for file placement and the currently assigned storage pool.

GPFS_ATTRFLAG_NO_PLACEMENT
Does not save attributes for file placement or the currently assigned storage pool.

GPFS_ATTRFLAG_IGNORE_POOL
Saves attributes for file placement but does not save the currently assigned storage pool.

GPFS_ATTRFLAG_USE_POLICY
Uses the restore policy rules to determine the pool ID.

GPFS_ATTRFLAG_INCL_DMAPI
Includes the DMAPI attributes.

GPFS_ATTRFLAG_FINALIZE_ATTRS
Finalizes immutability attributes.

GPFS_ATTRFLAG_SKIP_IMMUTABLE
Skips immutable attributes.

GPFS_ATTRFLAG_INCL_ENCR
Includes encryption attributes.

GPFS_ATTRFLAG_SKIP_CLONE
Skips clone attributes.

GPFS_ATTRFLAG_MODIFY_CLONEPARENT
Allows modification on the clone parent.

bufferP
Pointer to a buffer to store the extended attribute information.

bufferSize
The size of the buffer that was passed in.

attrSizeP
If successful, returns the actual size of the attribute information that was stored in the buffer. If the `bufferSize` was too small, returns the minimum buffer size.

Exit status

If the `gpfs_fgetattrs()` subroutine is successful, it returns a value of 0.

If the `gpfs_fgetattrs()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EBADF
The file descriptor is not valid.

EFAULT
The address is not valid.

EINVAL
The file descriptor does not refer to a GPFS file.

ENOSPC
`bufferSize` is too small to return all of the attributes. On return, `attrSizeP` is set to the required size.

ENOSYS
The `gpfs_fgetattrs()` subroutine is not supported under the current file system format.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_fputattrs() subroutine

Sets all the extended file attributes for a file.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_fputattrs(gpfs_file_t fileDesc,
                   int flags,
                   void *bufferP);
```

Description

The `gpfs_fputattrs()` subroutine, together with `gpfs_fgetattrs()`, is intended for use by a backup program to save (`gpfs_fgetattrs()`) and restore (`gpfs_fputattrs()`) all of the extended attributes of a file. This subroutine also sets the storage pool for the file and sets data replication to the values that are saved in the extended attributes.

If the saved storage pool is not valid or if the `GPFS_ATTRFLAG_IGNORE_POOL` flag is set, GPFS will select the storage pool by matching a **PLACEMENT** rule using the saved file attributes. If GPFS fails to match a placement rule or if there are no placement rules installed, GPFS assigns the file to the system storage pool.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fileDesc
The file descriptor identifying the file whose extended attributes are being set.

flags
Must have one of the following values:

GPFS_ATTRFLAG_DEFAULT
Restores the previously assigned storage pool and previously assigned data replication.

GPFS_ATTRFLAG_NO_PLACEMENT
Does not change storage pool and data replication.

GPFS_ATTRFLAG_IGNORE_POOL
Selects storage pool and data replication by matching the saved attributes to a placement rule instead of restoring the saved storage pool.

GPFS_ATTRFLAG_USE_POLICY
Uses the restore policy rules to determine the pool ID.

GPFS_ATTRFLAG_INCL_DMAPI
Includes the DMAPI attributes.

GPFS_ATTRFLAG_FINALIZE_ATTRS
Finalizes immutability attributes.

GPFS_ATTRFLAG_SKIP_IMMUTABLE
Skips immutable attributes.

GPFS_ATTRFLAG_INCL_ENCR
Includes encryption attributes.

GPFS_ATTRFLAG_SKIP_CLONE
Skips clone attributes.

GPFS_ATTRFLAG_MODIFY_CLONEPARENT
Allows modification on the clone parent.

Non-placement attributes such as ACLs are always restored, regardless of value of the flag.

bufferP
A pointer to the buffer containing the extended attributes for the file.

If you specify a value of NULL, all extended ACLs for the file are deleted.

Exit status

If the `gpfs_fputattrs()` subroutine is successful, it returns a value of 0.

If the `gpfs_fputattrs()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EBADF
The file descriptor is not valid.

EINVAL
The buffer pointed to by `bufferP` does not contain valid attribute data, or the file descriptor does not refer to a GPFS file.

ENOSYS
The `gpfs_fputattrs()` subroutine is not supported under the current file system format.

Examples

To copy extended file attributes from file `f1` to file `f2`:

```
char buf[4096];
int f1, f2, attrSize, rc;

rc = gpfs_fgetattrs(f1, GPFS_ATTRFLAG_DEFAULT, buf, sizeof(buf), &attrSize);
if (rc != 0)
   ...                                     // error handling
if (attrSize != 0)
   rc = gpfs_fputattrs(f2, 0, buf);        // copy attributes from f1 to f2
else
   rc = gpfs_fputattrs(f2, 0, NULL);       // f1 has no attributes
                                           // delete attributes on f2
```

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_fputattrswithpathname() subroutine

Sets all of the extended file attributes for a file and invokes the policy engine for RESTORE rules.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_fputattrswithpathname(gpfs_file_t fileDesc,
                               int flags,
                               void *bufferP,
                               const char *pathName);
```

Description

The `gpfs_fputattrswithpathname()` subroutine sets all of the extended attributes of a file. In addition, `gpfs_fputattrswithpathname()` invokes the policy engine using the saved attributes to match a **RESTORE** rule to set the storage pool and the data replication for the file. The caller should include the full path to the file (including the file name) to allow rule selection based on file name or path. If the file fails to match a **RESTORE** rule or if there are no **RESTORE** rules installed, GPFS selects the storage pool and data replication as it does when calling `gpfs_fputattrs()`.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from one the following libraries:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fileDesc
Is the file descriptor that identifies the file whose extended attributes are to be set.

flags
Must have one of the following values:

GPFS_ATTRFLAG_DEFAULT
Uses the saved attributes to match a **RESTORE** rule to set the storage pool and the data replication for the file.

GPFS_ATTRFLAG_NO_PLACEMENT
Does not change storage pool and data replication.

GPFS_ATTRFLAG_IGNORE_POOL
Checks the file to see if it matches a **RESTORE** rule. If the file fails to match a **RESTORE** rule, GPFS ignores the saved storage pool and selects a pool by matching the saved attributes to a **PLACEMENT** rule.

GPFS_ATTRFLAG_USE_POLICY
Uses the restore policy rules to determine the pool ID.

GPFS_ATTRFLAG_INCL_DMAPI
Includes the DMAPI attributes.

GPFS_ATTRFLAG_FINALIZE_ATTRS
Finalizes immutability attributes.

GPFS_ATTRFLAG_SKIP_IMMUTABLE
Skips immutable attributes.

GPFS_ATTRFLAG_INCL_ENCR
Includes encryption attributes.

GPFS_ATTRFLAG_SKIP_CLONE
Skips clone attributes.

GPFS_ATTRFLAG_MODIFY_CLONEPARENT
Allows modification on the clone parent.

Non-placement attributes such as ACLs are always restored, regardless of value of the flag.

bufferP
A pointer to the buffer containing the extended attributes for the file.

If you specify a value of NULL, all extended ACLs for the file are deleted.

pathName
A pointer to the path name to a file or directory.

Exit status

If the `gpfs_fputattrswithpathname()` subroutine is successful, it returns a value of 0.

If the `gpfs_fputattrswithpathname()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EBADF
The file descriptor is not valid.

EINVAL
The buffer to which `bufferP` points does not contain valid attribute data.

ENOENT
No such file or directory.

ENOSYS
The `gpfs_fputattrswithpathname()` subroutine is not supported under the current file system format.

Examples

Refer to "gpfs_fputattrs() subroutine" on page 867 for examples.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_free_fssnaphandle() subroutine

Frees a GPFS file system snapshot handle.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
void gpfs_free_fssnaphandle(gpfs_fssnap_handle_t *fssnapHandle);
```

Description

The `gpfs_free_fssnaphandle()` subroutine frees the snapshot handle that is passed. The return value is always void.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fssnapHandle
File system snapshot handle.

Exit status

The `gpfs_free_fssnaphandle()` subroutine always returns void.

Exceptions

None.

Error status

None.

Examples

For an example using `gpfs_free_fssnaphandle()`, see `/usr/lpp/mmfs/samples/util/tstimes.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_fssnap_handle_t structure

Contains a handle for a GPFS file system or snapshot.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Structure

```
typedef struct gpfs_fssnap_handle gpfs_fssnap_handle_t;
```

Description

A file system or snapshot is uniquely identified by an `fssnapId` of type `gpfs_fssnap_id_t`. While the `fssnapId` is permanent and global, a shorter `fssnapHandle` is used by the backup application programming interface to identify the file system and snapshot being accessed. The `fssnapHandle`, like a POSIX file descriptor, is volatile and may be used only by the program that created it.

There are three ways to create a file system snapshot handle:

1. By using the name of the file system and snapshot
2. By specifying the path through the mount point
3. By providing an existing file system snapshot ID

Additional subroutines are provided to obtain the permanent, global `fssnapId` from the `fssnapHandle`, or to obtain the path or the names for the file system and snapshot, if they are still available in the file system.

The file system must be mounted in order to use the backup programming application interface. If the `fssnapHandle` is created by the path name, the path may be relative and may specify any file or directory in the file system. Operations on a particular snapshot are indicated with a path to a file or directory within that snapshot. If the `fssnapHandle` is created by name, the file system's unique name may be specified (for example, `fs1`) or its device name may be provided (for example, `/dev/fs1`). To specify an operation on the active file system, the pointer to the snapshot's name should be set to NULL or a zero-length string provided.

The name of the directory under which all snapshots appear may be obtained by the `gpfs_get_snapdirname()` subroutine. By default this is `.snapshots`, but it can be changed using the `mmsnapdir` command. The `gpfs_get_snapdirname()` subroutine returns the currently set value, which is the one that was last set by the `mmsnapdir` command, or the default, if it was never changed.

Members

gpfs_fssnap_handle
File system snapshot handle

Examples

For an example using `gpfs_fssnap_handle_t`, see `/usr/lpp/mmfs/samples/util/tsgetusage.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_fssnap_id_t structure

Contains a permanent identifier for a GPFS file system or snapshot.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Structure

```
typedef struct gpfs_fssnap_id
{
  char opaque[48];
} gpfs_fssnap_id_t;
```

Description

A file system or snapshot is uniquely identified by an `fssnapId` of type `gpfs_fssnap_id_t`. The `fssnapId` is a permanent and global identifier that uniquely identifies an active file system or a read-only snapshot of a file system. Every snapshot of a file system has a unique identifier that is also different from the identifier of the active file system itself.

The `fssnapId` is obtained from an open `fssnapHandle`. Once obtained, the `fssnapId` should be stored along with the file system's data for each backup. The `fssnapId` is required to generate an incremental backup. The `fssnapId` identifies the previously backed up file system or snapshot and allows the inode scan to return only the files and data that have changed since that previous scan.

Members

opaque
A 48 byte area for containing the snapshot identifier.

Examples

For an example using `gpfs_fssnap_id_t`, see `/usr/lpp/mmfs/samples/util/tsinode.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_fstat() subroutine

Returns exact file status for a GPFS file.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_fstat(gpfs_file_t fileDesc,
               gpfs_stat64_t *buffer);
```

Description

The `gpfs_fstat()` subroutine is used to obtain exact information about the file associated with the `fileDesc` parameter. This subroutine is provided as an alternative to the `stat()` subroutine, which may not provide exact `mtime` and `atime` values. For more information, see the topic *Exceptions to Open Group technical standards* in the *IBM Spectrum Scale: Administration Guide*.

`read`, `write`, or `execute` permission for the named file is not required, but all directories listed in the path leading to the file must be searchable. The file information is written to the area specified by the `buffer` parameter.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fileDesc
The file descriptor identifying the file for which exact status information is requested.

buffer
A pointer to the `gpfs_stat64_t` structure in which the information is returned. The `gpfs_stat64_t` structure is described in the `sys/stat.h` file.

Exit status

If the `gpfs_fstat()` subroutine is successful, it returns a value of 0.

If the `gpfs_fstat()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EBADF
The file descriptor is not valid.

EINVAL
The file descriptor does not refer to a GPFS file or a regular file.

ENOSYS
The `gpfs_fstat()` subroutine is not supported under the current file system format.

ESTALE
The cached file system information was not valid.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_fstat_x() subroutine

Returns extended status information for a GPFS file with specified accuracy.

Library

GPFS library. Runs on AIX, Linux, and Windows.

Synopsis

```
#include <gpfs.h>
int gpfs_fstat_x(gpfs_file_t fileDesc,
            unsigned int *st_litemask,
            gpfs_iattr64_t *iattr,
            size_t iattrBufLen);
```

Description

The `gpfs_fstat_x()` subroutine is similar to the `gpfs_fstat()` subroutine but returns more information in a `gpfs_iattr64` structure that is defined in `gpfs.h`. This subroutine is supported only on the Linux operating system.

Your program must verify that the version of the `gpfs_iattr64` structure that is returned in the field `ia_version` is the same as the version that you are using. Versions are defined in `gpfs.h` with the pattern `GPFS_IA64_VERSION*`.

File permissions `read`, `write`, and `execute` are not required for the specified file, but all the directories in the specified path must be searchable.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.so` for Linux

Parameters

gpfs_file_t fileDesc
The file descriptor of a file for which information is requested.

***st_litemask**
A pointer to a bitmask specification of the items that you want to be returned exactly. Bitmasks are defined in `gpfs.h` with the pattern `GPFS_SLITE_*BIT`. The subroutine returns exact values for the items that you specify in the bitmask. The subroutine also sets bits in the bitmask to indicate any other items that are exact.

***iattr**
A pointer to a `gpfs_iattr64_t` structure in which the information is returned. The structure is described in the `gpfs.h` file.

iattrBufLen
The length of your `gpfs_iattr64_t` structure, as given by `sizeof(myStructure)`. The subroutine does not write data past this limit. The field `ia_reclen` in the returned structure is the length of the `gpfs_iattr64_t` structure that the subroutine is using.

Exit status

If the `gpfs_fstat_x()` subroutine is successful, it returns a value of 0.

If the `gpfs_fstat_x()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following errors:

EBADF
The file handle does not refer to an existing or accessible object.

ENOSYS
The `gpfs_fstat_x()` subroutine is not supported under the current file system format.

ESTALE
The cached file system information was invalid.

Location

`/usr/lpp/mmfs/lib`

gpfs_get_fsname_from_fssnaphandle() subroutine

Obtains a file system name from its snapshot handle.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
const char *gpfs_get_fsname_from_fssnaphandle(gpfs_fssnap_handle_t *fssnapHandle);
```

Description

The `gpfs_get_fsname_from_fssnaphandle()` subroutine returns a pointer to the name of file system that is uniquely identified by the file system snapshot handle.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fssnapHandle
File system snapshot handle.

Exit status

If the `gpfs_get_fsname_from_fssnaphandle()` subroutine is successful, it returns a pointer to the name of the file system identified by the file system snapshot handle.

If the `gpfs_get_fsname_from_fssnaphandle()` subroutine is unsuccessful, it returns NULL and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

ENOSYS
The `gpfs_get_fsname_from_fssnaphandle()` subroutine is not available.

EPERM
The caller does not have superuser privileges.

GPFS_E_INVAL_FSSNAPHANDLE
The file system snapshot handle is not valid.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_get_fssnaphandle_by_fssnapid() subroutine

Obtains a file system snapshot handle using its snapshot ID.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
gpfs_fssnap_handle_t *gpfs_get_fssnaphandle_by_fssnapid(const gpfs_fssnap_id_t *fssnapId);
```

Description

The `gpfs_get_fssnaphandle_by_fssnapid()` subroutine creates a handle for the file system or snapshot that is uniquely identified by the permanent, unique snapshot ID.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fssnapId
File system snapshot ID

Exit status

If the `gpfs_get_fssnaphandle_by_fssnapid()` subroutine is successful, it returns a pointer to the file system snapshot handle.

If the `gpfs_get_fssnaphandle_by_fssnapid()` subroutine is unsuccessful, it returns NULL and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

ENOMEM
Space could not be allocated for the file system snapshot handle.

ENOSYS
The `gpfs_get_fssnaphandle_by_fssnapid()` subroutine is not available.

EPERM
The caller does not have superuser privileges.

GPFS_E_INVAL_FSSNAPID
The file system snapshot ID is not valid.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_get_fssnaphandle_by_name() subroutine

Obtains a file system snapshot handle using its name.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
gpfs_fssnap_handle_t *gpfs_get_fssnaphandle_by_name(const char *fsName,
                                                    const char *snapName);
```

Description

The `gpfs_get_fssnaphandle_by_name()` subroutine creates a handle for the file system or snapshot that is uniquely identified by the file system's name and the name of the snapshot.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fsName
A pointer to the name of the file system whose snapshot handle is desired.

snapName
A pointer to the name of the snapshot whose snapshot handle is desired, or NULL to access the active file system rather than a snapshot within the file system.

Exit status

If the `gpfs_get_fssnaphandle_by_name()` subroutine is successful, it returns a pointer to the file system snapshot handle.

If the `gpfs_get_fssnaphandle_by_name()` subroutine is unsuccessful, it returns NULL and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

ENOENT
The file system name is not valid.

ENOMEM
Space could not be allocated for the file system snapshot handle.

ENOSYS
The `gpfs_get_fssnaphandle_by_name()` subroutine is not available.

EPERM
The caller does not have superuser privileges.

GPFS_E_INVAL_SNAPNAME
The snapshot name is not valid.

Examples

For an example using `gpfs_get_fssnaphandle_by_name()`, see `/usr/lpp/mmfs/samples/util/tsinode.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_get_fssnaphandle_by_path() subroutine

Obtains a file system snapshot handle using its path name.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
gpfs_fssnap_handle_t *gpfs_get_fssnaphandle_by_path(const char *pathName);
```

Description

The `gpfs_get_fssnaphandle_by_path()` subroutine creates a handle for the file system or snapshot that is uniquely identified by a path through the file system's mount point to a file or directory within the file system or snapshot.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

pathName
A pointer to the path name to a file or directory within the desired file system or snapshot.

Exit status

If the `gpfs_get_fssnaphandle_by_path()` subroutine is successful, it returns a pointer to the file system snapshot handle.

If the `gpfs_get_fssnaphandle_by_path()` subroutine is unsuccessful, it returns NULL and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

ENOENT
The path name is not valid.

ENOMEM
Space could not be allocated for the file system snapshot handle.

ENOSYS
The `gpfs_get_fssnaphandle_by_path()` subroutine is not available.

EPERM
The caller does not have superuser privileges.

Examples

For an example using `gpfs_get_fssnaphandle_by_path()`, see `/usr/lpp/mmfs/samples/util/tsgetusage.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_get_fssnapid_from_fssnaphandle() subroutine

Obtains a file system snapshot ID using its handle.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_get_fssnapid_from_fssnaphandle(gpfs_fssnap_handle_t *fssnapHandle,
                                        gpfs_fssnap_id_t *fssnapId);
```

Description

The `gpfs_get_fssnapid_from_fssnaphandle()` subroutine obtains the permanent, globally unique file system snapshot ID of the file system or snapshot identified by the open file system snapshot handle.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fssnapHandle
File system snapshot handle.

fssnapId
File system snapshot ID.

Exit status

If the `gpfs_get_fssnapid_from_fssnaphandle()` subroutine is successful, it returns a pointer to the file system snapshot ID.

If the `gpfs_get_fssnapid_from_fssnaphandle()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EFAULT
Size mismatch for `fssnapId`.

EINVAL
NULL pointer given for returned `fssnapId`.

ENOSYS
The `gpfs_get_fssnapid_from_fssnaphandle()` subroutine is not available.

GPFS_E_INVAL_FSSNAPHANDLE
The file system snapshot handle is not valid.

Examples

For an example using `gpfs_get_fssnapid_from_fssnaphandle()`, see `/usr/lpp/mmfs/samples/util/tsinode.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_get_pathname_from_fssnaphandle() subroutine

Obtains a file system path name using its snapshot handle.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
const char *gpfs_get_pathname_from_fssnaphandle(gpfs_fssnap_handle_t *fssnapHandle);
```

Description

The `gpfs_get_pathname_from_fssnaphandle()` subroutine obtains the path name of the file system or snapshot identified by the open file system snapshot handle.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fssnapHandle
File system snapshot handle.

Exit status

If the `gpfs_get_pathname_from_fssnaphandle()` subroutine is successful, it returns a pointer to the path name of the file system or snapshot.

If the `gpfs_get_pathname_from_fssnaphandle()` subroutine is unsuccessful, it returns NULL and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

ENOSYS
The `gpfs_get_pathname_from_fssnaphandle()` subroutine is not available.

EPERM
The caller does not have superuser privileges.

GPFS_E_INVAL_FSSNAPHANDLE
The file system snapshot handle is not valid.

Examples

For an example using `gpfs_get_pathname_from_fssnaphandle()`, see `/usr/lpp/mmfs/samples/util/tsfindinode.c`.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX

/usr/lpp/mmfs/lib/libgpfs.so for Linux

gpfs_get_snapdirname() subroutine

Obtains the name of the directory containing global snapshots.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_get_snapdirname(gpfs_fssnap_handle_t *fssnapHandle,
                         char *snapdirName,
                         int bufLen);
```

Description

The `gpfs_get_snapdirname()` subroutine obtains the name of the directory that contains global snapshots.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fssnapHandle
File system snapshot handle.

snapdirName
Buffer into which the name of the snapshot directory will be copied.

bufLen
The size of the provided buffer.

Exit status

If the `gpfs_get_snapdirname()` subroutine is successful, it returns a value of 0 and the `snapdirName` and `bufLen` parameters are set.

If the `gpfs_get_snapdirname()` subroutine is unsuccessful, it returns a value of -1 and the global error variable `errno` is set to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

ENOMEM
Unable to allocate memory for the request.

ENOSYS
The `gpfs_get_snapdirname()` subroutine is not available.

EPERM
The caller does not have superuser privileges.

ERANGE
The buffer is too small to return the snapshot directory name.

ESTALE
The cached file system information was not valid.

GPFS_E_INVAL_FSSNAPHANDLE
The file system snapshot handle is not valid.

E2BIG
The buffer is too small to return the snapshot directory name.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_get_snapname_from_fssnaphandle() subroutine

Obtains a snapshot name using its file system snapshot handle.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
const char *gpfs_get_snapname_from_fssnaphandle(gpfs_fssnap_handle_t *fssnapHandle);
```

Description

The `gpfs_get_snapname_from_fssnaphandle()` subroutine obtains a pointer to the name of a GPFS snapshot given its file system snapshot handle. If the `fssnapHandle` identifies an active file system, as opposed to a snapshot of a file system, `gpfs_get_snapname_from_fssnaphandle()` returns a pointer to a zero-length snapshot name and a successful return code.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fssnapHandle
File system snapshot handle.

Exit status

If the `gpfs_get_snapname_from_fssnaphandle()` subroutine is successful, it returns a pointer to the name of the snapshot.

If the `gpfs_get_snapname_from_fssnaphandle()` subroutine is unsuccessful, it returns NULL and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

ENOSYS
The `gpfs_get_snapname_from_fssnaphandle()` subroutine is not available.

EPERM
The caller does not have superuser privileges.

GPFS_E_INVAL_FSSNAPHANDLE
The file system snapshot handle is not valid.

GPFS_E_INVAL_SNAPNAME
The snapshot has been deleted.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX

/usr/lpp/mmfs/lib/libgpfs.so for Linux

gpfs_getacl() subroutine

Retrieves the access control information for a GPFS file.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_getacl(const char *pathname,
                int flags,
                void *acl);
```

Description

The `gpfs_getacl()` subroutine, together with the `gpfs_putacl()` subroutine, is intended for use by a backup program to save (`gpfs_getacl()`) and restore (`gpfs_putacl()`) the ACL information for the file.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

pathname
The path identifying the file for which the ACLs are being obtained.

flags
Consists of one of these values:

0
Indicates that the `acl` parameter is to be mapped with the `gpfs_opaque_acl_t` structure.

The `gpfs_opaque_acl_t` structure should be used by backup and restore programs.

GPFS_GETACL_STRUCT
Indicates that the `acl` parameter is to be mapped with the `gpfs_acl_t` structure.

The `gpfs_acl_t` structure is provided for applications that need to interpret the ACL.

acl
Pointer to a buffer mapped by the structure `gpfs_opaque_acl_t` or `gpfs_acl_t`, depending on the value of `flags`.

The first four bytes of the buffer must contain its total size.

Exit status

If the `gpfs_getacl()` subroutine is successful, it returns a value of 0.

If the `gpfs_getacl()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EINVAL
The path name does not refer to a GPFS file or a regular file.

ENOMEM
Unable to allocate memory for the request.

ENOTDIR
File is not a directory.

ENOSPC
The buffer is too small to return the entire ACL. The required buffer size is returned in the first four bytes of the buffer pointed to by `acl`.

ENOSYS
The `gpfs_getacl()` subroutine is not supported under the current file system format.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_getacl_fd() subroutine

Retrieves the access control information for a GPFS file.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_getacl_fd(gpfs_file_t fileDesc,
                   int flags,
                   void *acl);
```

Description

The `gpfs_getacl_fd()` subroutine together with the `gpfs_putacl_fd()` subroutine, is intended for use by a backup program to save (`gpfs_getacl_fd()`) and restore (`gpfs_putacl_fd()`) the ACL information for the file.

Note: Compile any program that uses this subroutine with the `lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fileDesc
A file descriptor that identifies the file for which the ACLs are being obtained.

flags
Consists of one of these values:

0
Indicates that the `acl` parameter is to be mapped with the `gpfs_opaque_acl_t` structure.

The `gpfs_opaque_acl_t` structure is used by backup and restore programs.

GPFS_GETACL_STRUCT
Indicates that the `acl` parameter is to be mapped with the `gpfs_acl_t` structure.

The `gpfs_acl_t` structure is provided for applications that need to interpret the ACL.

acl
Pointer to a buffer mapped by the structure `gpfs_opaque_acl_t` or `gpfs_acl_t`, depending on the value of `flags`.

The first four bytes of the buffer must contain its total size.

The `gpfs_opaque_acl_t` structure contains size, version, and ACL type information for the `gpfs_getacl()` and `gpfs_putacl()` subroutines.

Exit status

If the `gpfs_getacl_fd()` subroutine is successful, it returns a value of 0.

If the `gpfs_getacl_fd()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following list:

Error codes include but are not limited to the following:

EINVAL
The file descriptor does not refer to a GPFS file or a regular file.

EBADF
The file descriptor is not valid.

ENOMEM
Unable to allocate memory for the request.

ENOTDIR
File is not a directory.

ENOSPC
The buffer is too small to return the entire ACL. The required buffer size is returned in the first four bytes of the buffer pointed to by `acl`.

ENOSYS
The `gpfs_getacl_fd()` subroutine is not supported under the current file system format.

Location

`/usr/lpp/mmfs/lib/l ibgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_iattr_t structure

Contains attributes of a GPFS inode.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Structure

```
typedef struct gpfs_iattr
{
  int              ia_version;    /* this struct version */
  int              ia_reclen;     /* sizeof this structure */
  int              ia_checksum;   /* validity check on iattr struct */
  gpfs_mode_t      ia_mode;       /* access mode */
  gpfs_uid_t       ia_uid;        /* owner uid */
  gpfs_gid_t       ia_gid;        /* owner gid */
  gpfs_ino_t       ia_inode;      /* file inode number */
  gpfs_gen_t       ia_gen;        /* inode generation number */
  gpfs_nlink_t     ia_nlink;      /* number of links */
  short            ia_flags;      /* Flags (defined below) */
  int              ia_blocksize;  /* preferred block size for io */
  gpfs_mask_t      ia_mask;       /* Initial attribute mask (not used) */
  unsigned int     ia_pad1;       /* reserved space */
  gpfs_off64_t     ia_size;       /* file size in bytes */
  gpfs_off64_t     ia_blocks;     /* 512 byte blocks of disk held by file */
  gpfs_timestruc_t ia_atime;      /* time of last access */
  gpfs_timestruc_t ia_mtime;      /* time of last data modification */
  gpfs_timestruc_t ia_ctime;      /* time of last status change */
  gpfs_dev_t       ia_rdev;       /* id of device */
  unsigned int     ia_xperm;      /* extended attributes (defined below) */
  unsigned int     ia_modsnapid;  /* snapshot id of last modification */
  unsigned int     ia_filesetid;  /* fileset ID */
  unsigned int     ia_datapoolid; /* storage pool ID for data */
  unsigned int     ia_pad2;       /* reserved space */
} gpfs_iattr_t;

/* Define flags for inode attributes */
#define GPFS_IAFLAG_SNAPDIR         0x0001 /* (obsolete) */
#define GPFS_IAFLAG_USRQUOTA        0x0002 /* inode is a user quota file */
#define GPFS_IAFLAG_GRPQUOTA        0x0004 /* inode is a group quota file */
#define GPFS_IAFLAG_ERROR           0x0008 /* error reading inode */
/* Define flags for inode replication attributes */
#define GPFS_IAFLAG_FILESET_ROOT    0x0010 /* root dir of a fileset */
#define GPFS_IAFLAG_NO_SNAP_RESTORE 0x0020 /* don't restore from snapshots */
#define GPFS_IAFLAG_FILESETQUOTA    0x0040 /* inode is a fileset quota file */
#define GPFS_IAFLAG_COMANAGED       0x0080 /* file data is co-managed */
#define GPFS_IAFLAG_ILLPLACED       0x0100 /* may not be properly placed */
#define GPFS_IAFLAG_REPLMETA        0x0200 /* metadata replication set */
#define GPFS_IAFLAG_REPLDATA        0x0400 /* data replication set */
#define GPFS_IAFLAG_EXPOSED         0x0800 /* may have data on suspended disks */
#define GPFS_IAFLAG_ILLREPLICATED   0x1000 /* may not be properly replicated */
#define GPFS_IAFLAG_UNBALANCED      0x2000 /* may not be properly balanced */
#define GPFS_IAFLAG_DATAUPDATEMISS  0x4000 /* has stale data blocks on
                                              unavailable disk */
#define GPFS_IAFLAG_METAUPDATEMISS  0x8000 /* has stale metadata on
                                              unavailable disk */

#define GPFS_IAFLAG_IMMUTABLE       0x00010000 /* Immutability */
#define GPFS_IAFLAG_INDEFRETENT     0x00020000 /* Indefinite retention */
#define GPFS_IAFLAG_SECUREDELETE    0x00040000 /* Secure deletion */

#define GPFS_IAFLAG_TRUNCMANAGED    0x00080000 /* dmapi truncate event enabled */
#define GPFS_IAFLAG_READMANAGED     0x00100000 /* dmapi read event enabled */
#define GPFS_IAFLAG_WRITEMANAGED    0x00200000 /* dmapi write event enabled */

#define GPFS_IAFLAG_APPENDONLY      0x00400000 /* AppendOnly only */
#define GPFS_IAFLAG_DELETED         0x00800000 /* inode has been deleted */

/* Define flags for extended attributes */
#define GPFS_IAXPERM_ACL            0x0001 /* file has acls */
#define GPFS_IAXPERM_XATTR          0x0002 /* file has extended attributes */
#define GPFS_IAXPERM_DMATTR         0x0004 /* file has dm attributes */
```

```
#define GPFS_IAXPERM_DOSATTR        0x0008 /* file has non-default dos attrs */
#define GPFS_IAXPERM_RPATTR         0x0010 /* file has restore policy attrs */
```

Description

The gpfs_iattr_t structure contains the various attributes of a GPFS inode.

Members

ia_version
The version number of this structure.

ia_reclen
The size of this structure.

ia_checksum
The checksum for this gpfs_iattr structure.

ia_mode
The access mode for this inode.

ia_uid
The owner user ID for this inode.

ia_gid
The owner group ID for this inode.

ia_inode
The file inode number.

ia_gen
The inode generation number.

ia_nlink
The number of links for this inode.

ia_flags
The flags defined for inode attributes.

ia_blocksize
The preferred block size for I/O.

ia_mask
The initial attribute mask (not used).

ia_pad1
Reserved space.

ia_size
The file size in bytes.

ia_blocks
The number of 512 byte blocks of disk held by the file.

ia_atime
The time of last access.

ia_mtime
The time of last data modification.

ia_ctime
The time of last status change.

ia_rdev
The ID of the device.

ia_xperm
Indicator - nonzero if file has extended ACL.

ia_modsnapid
Internal snapshot ID indicating the last time that the file was modified. Internal snapshot IDs for the current snapshots are displayed by the `mmlssnapshot` command.

ia_filesetid
The fileset ID for the inode.

ia_datapoolid
The storage pool ID for data for the inode.

ia_pad2
Reserved space.

Examples

For an example using `gpfs_iattr_t`, see `/usr/lpp/mmfs/samples/util/tsinode.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_iattr64_t structure

Contains attributes of a GPFS inode.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Structure

```
typedef struct gpfs_iattr64
{
  int                ia_version;    /* this struct version */
  int                ia_reclen;     /* sizeof this structure */
  int                ia_checksum;   /* validity check on iattr struct */
  gpfs_mode_t        ia_mode;       /* access mode */
  gpfs_uid64_t       ia_uid;        /* owner uid */
  gpfs_gid64_t       ia_gid;        /* owner gid */
  gpfs_ino64_t       ia_inode;      /* file inode number */
  gpfs_gen64_t       ia_gen;        /* inode generation number */
  gpfs_nlink64_t     ia_nlink;      /* number of links */
  gpfs_off64_t       ia_size;       /* file size in bytes */
  gpfs_off64_t       ia_blocks;     /* 512 byte blocks of disk held by file */
  gpfs_timestruc64_t ia_atime;      /* time of last access */
  unsigned int       ia_winflags;   /* windows flags (defined below) */
  unsigned int       ia_pad1;       /* reserved space */
  gpfs_timestruc64_t ia_mtime;      /* time of last data modification */
  unsigned int       ia_flags;      /* flags (defined below) */
 /* next four bytes were ia_pad2 */
  unsigned char      ia_repl_data;  /* data replication factor */
  unsigned char      ia_repl_data_max; /* data replication max factor */
  unsigned char      ia_repl_meta;  /* meta data replication factor */
  unsigned char      ia_repl_meta_max; /* meta data replication max factor */
  gpfs_timestruc64_t ia_ctime;      /* time of last status change */
  int                ia_blocksize;  /* preferred block size for io */
  unsigned int       ia_pad3;       /* reserved space */
  gpfs_timestruc64_t ia_createtime; /* creation time */
  gpfs_mask_t        ia_mask;       /* initial attribute mask (not used) */
  int                ia_pad4;       /* reserved space */
  unsigned int       ia_reserved[GPFS_IA64_RESERVED]; /* reserved space */
  unsigned int       ia_xperm;      /* extended attributes (defined below) */
  gpfs_dev_t         ia_dev;        /* id of device containing file */
  gpfs_dev_t         ia_rdev;       /* device id (if special file) */
  unsigned int       ia_pcacheflags; /* pcache inode bits */
  gpfs_snapid64_t    ia_modsnapid;  /* snapshot id of last modification */
  unsigned int       ia_filesetid;  /* fileset ID */
  unsigned int       ia_datapoolid; /* storage pool ID for data */
  gpfs_ino64_t       ia_inode_space_mask; /* inode space mask of this file system */
                                          /* This value is saved in the iattr structure
                                             during backup and used during restore */

  gpfs_off64_t       ia_dirminsize; /* dir pre-allocation size in bytes */
  unsigned int       ia_unused[GPFS_IA64_UNUSED];  /* reserved space */
} gpfs_iattr64_t;

#ifdef GPFS_64BIT_INODES
  #undef  GPFS_IA_VERSION
  #define GPFS_IA_VERSION GPFS_IA_VERSION64
  #define gpfs_iattr_t gpfs_iattr64_t
#endif


/* Define flags for inode attributes */
#define GPFS_IAFLAG_SNAPDIR         0x0001 /* (obsolete) */
#define GPFS_IAFLAG_USRQUOTA        0x0002 /* inode is a user quota file */
#define GPFS_IAFLAG_GRPQUOTA        0x0004 /* inode is a group quota file */
#define GPFS_IAFLAG_ERROR           0x0008 /* error reading inode */
/* Define flags for inode replication attributes */
#define GPFS_IAFLAG_FILESET_ROOT    0x0010 /* root dir of a fileset */
#define GPFS_IAFLAG_NO_SNAP_RESTORE 0x0020 /* don't restore from snapshots */
#define GPFS_IAFLAG_FILESETQUOTA    0x0040 /* inode is a fileset quota file */
#define GPFS_IAFLAG_COMANAGED       0x0080 /* file data is co-managed */
#define GPFS_IAFLAG_ILLPLACED       0x0100 /* may not be properly placed */
#define GPFS_IAFLAG_REPLMETA        0x0200 /* metadata replication set */
#define GPFS_IAFLAG_REPLDATA        0x0400 /* data replication set */
```

```
#define GPFS_IAFLAG_EXPOSED         0x0800 /* may have data on suspended disks */
#define GPFS_IAFLAG_ILLREPLICATED   0x1000 /* may not be properly replicated */
#define GPFS_IAFLAG_UNBALANCED      0x2000 /* may not be properly balanced */
#define GPFS_IAFLAG_DATAUPDATEMISS  0x4000 /* has stale data blocks on
                                              unavailable disk */
#define GPFS_IAFLAG_METAUPDATEMISS  0x8000 /* has stale metadata on
                                              unavailable disk */

#define GPFS_IAFLAG_IMMUTABLE       0x00010000 /* Immutability */
#define GPFS_IAFLAG_INDEFRETENT     0x00020000 /* Indefinite retention */
#define GPFS_IAFLAG_SECUREDELETE    0x00040000 /* Secure deletion */

#define GPFS_IAFLAG_TRUNCMANAGED    0x00080000 /* dmapi truncate event enabled */
#define GPFS_IAFLAG_READMANAGED     0x00100000 /* dmapi read event enabled */
#define GPFS_IAFLAG_WRITEMANAGED    0x00200000 /* dmapi write event enabled */

#define GPFS_IAFLAG_APPENDONLY      0x00400000 /* AppendOnly only */
#define GPFS_IAFLAG_DELETED         0x00800000 /* inode has been deleted */
#ifdef ZIP
#define GPFS_IAFLAG_ILLCOMPRESSED   0x01000000 /* may not be properly compressed */
#endif
#define GPFS_IAFLAG_FPOILLPLACED    0x02000000 /* may not be properly placed per
                                                  FPO attributes (bgf, wad, wadfg) */
/* Define flags for window's attributes */
#define GPFS_IWINFLAG_ARCHIVE       0x0001 /* Archive */
#define GPFS_IWINFLAG_HIDDEN        0x0002 /* Hidden */
#define GPFS_IWINFLAG_NOTINDEXED    0x0004 /* Not content indexed */
#define GPFS_IWINFLAG_OFFLINE       0x0008 /* Off-line */
#define GPFS_IWINFLAG_READONLY      0x0010 /* Read-only */
#define GPFS_IWINFLAG_REPARSE       0x0020 /* Reparse point */
#define GPFS_IWINFLAG_SYSTEM        0x0040 /* System */
#define GPFS_IWINFLAG_TEMPORARY     0x0080 /* Temporary */
#define GPFS_IWINFLAG_COMPRESSED    0x0100 /* Compressed */
#define GPFS_IWINFLAG_ENCRYPTED     0x0200 /* Encrypted */
#define GPFS_IWINFLAG_SPARSE        0x0400 /* Sparse file */
#define GPFS_IWINFLAG_HASSTREAMS    0x0800 /* Has streams */

/* Define flags for extended attributes */
#define GPFS_IAXPERM_ACL            0x0001 /* file has acls */
#define GPFS_IAXPERM_XATTR          0x0002 /* file has extended attributes */
#define GPFS_IAXPERM_DMATTR         0x0004 /* file has dm attributes */
#define GPFS_IAXPERM_DOSATTR        0x0008 /* file has non-default dos attrs */
#define GPFS_IAXPERM_RPATTR         0x0010 /* file has restore policy attrs */

/* Define flags for pcache bits defined in the inode */
#define GPFS_ICAFLAG_CACHED   0x0001  /* "cached complete"  */
#define GPFS_ICAFLAG_CREATE   0x0002  /* "created"          */
#define GPFS_ICAFLAG_DIRTY    0x0004  /* "data dirty"       */
#define GPFS_ICAFLAG_LINK     0x0008  /* "hard linked"      */
#define GPFS_ICAFLAG_SETATTR  0x0010  /* "attr changed"     */
#define GPFS_ICAFLAG_LOCAL    0x0020  /* "local"            */
#define GPFS_ICAFLAG_APPEND   0x0040  /* "append"           */
#define GPFS_ICAFLAG_STATE    0x0080  /* "has remote state" */
```

Description

The `gpfs_iattr64_t` structure contains the various attributes of a GPFS inode.

Members

ia_version
The version number of this structure.

ia_reclen
The size of this structure.

ia_checksum
The checksum for this `gpfs_iattr64` structure.

ia_mode
The access mode for this inode.

ia_uid
The owner user ID for this inode.

ia_gid
The owner group ID for this inode.

ia_inode
The file inode number.

ia_gen
The inode generation number.

ia_nlink
The number of links for this inode.

ia_size
The file size in bytes.

ia_blocks
The number of 512 byte blocks of disk held by the file.

ia_atime
The time of last access.

ia_winflags
The Windows flags.

ia_pad1
Reserved space.

ia_mtime
The time of last data modification.

ia_flags
The flags defined for inode attributes.

ia_repl_data
The data replication factor.

ia_repl_data_max
The maximum data replication factor.

ia_repl_meta
The metadata replication factor.

ia_repl_meta_max
The maximum metadata replication factor.

ia_ctime
The time of last status change.

ia_blocksize
The preferred block size for I/O.

ia_pad3
Reserved space.

ia_createtime
The creation time.

ia_mask
The initial attribute mask (not used).

ia_pad4
Reserved space.

ia_reserved
Reserved space.

ia_xperm
Indicator - nonzero if file has extended ACL.

ia_dev
The ID of the device containing the file.

ia_rdev
The ID of the device.

ia_pcacheflags
The pcache inode bits.

ia_modsnapid
Internal snapshot ID indicating the last time that the file was modified. Internal snapshot IDs for the current snapshots are displayed by the `mmlssnapshot` command.

ia_filesetid
The fileset ID for the inode.

ia_datapoolid
The storage pool ID for data for the inode.

ia_dirminsize
Directory preallocation size in bytes.

ia_inode_space_mask
The inode space mask of this file system. This value is saved in the `iattr` structure during backup and used during restore.

ia_unused
Reserved space.

Examples

See the `gpfs_iattr_t` example in `/usr/lpp/mmfs/samples/util/tsinode.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_iclose() subroutine

Closes a file given its inode file handle.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
void gpfs_iclose(gpfs_ifile_t *ifile);
```

Description

The `gpfs_iclose()` subroutine closes an open file descriptor created by `gpfs_iopen()`.

For an overview of using `gpfs_iclose()` in a backup application, see the topic *Using APIs to develop backup applications* in the *IBM Spectrum Scale: Administration Guide*.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

ifile
Pointer to `gpfs_ifile_t` from `gpfs_iopen()`.

Exit status

The `gpfs_iclose()` subroutine returns void.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

ENOSYS
The `gpfs_iclose()` subroutine is not available.

EPERM
The caller does not have superuser privileges.

ESTALE
Cached file system information was not valid.

Examples

For an example using `gpfs_iclose()`, see `/usr/lpp/mmfs/samples/util/tsreaddir.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_ifile_t structure

Contains a handle for a GPFS inode.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Structure

```
typedef struct gpfs_ifile gpfs_ifile_t;
```

Description

The `gpfs_ifile_t` structure contains a handle for the file of a GPFS inode.

Members

gpfs_ifile
The handle for the file of a GPFS inode.

Examples

For an example using `gpfs_ifile_t`, see `/usr/lpp/mmfs/samples/util/tsfindinode.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_igetattrs() subroutine

Retrieves extended file attributes in opaque format.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_igetattrs(gpfs_ifile_t *ifile,
                   void *buffer,
                   int bufferSize,
                   int *attrSize);
```

Description

The `gpfs_igetattrs()` subroutine retrieves all extended file attributes in opaque format. This subroutine is intended for use by a backup program to save all extended file attributes (ACLs, attributes, and so forth). If the file does not have any extended attributes, the subroutine sets `attrSize` to zero.

Notes:

1. This call does not return extended attributes used for the Data Storage Management (XDSM) API (also known as DMAPI).
2. Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:
   - `libgpfs.a` for AIX
   - `libgpfs.so` for Linux

Parameters

ifile
Pointer to `gpfs_ifile_t` from `gpfs_iopen()`.

buffer
Pointer to buffer for returned attributes.

bufferSize
Size of the buffer.

attrSize
Pointer to returned size of attributes.

Exit status

If the `gpfs_igetattrs()` subroutine is successful, it returns a value of 0.

If the `gpfs_igetattrs()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

ENOSPC
The buffer is too small to return all attributes. Field `attrSize` will be set to the size necessary.

ENOSYS
The `gpfs_igetattrs()` subroutine is not available.

EPERM
The caller does not have superuser privileges.

ESTALE
Cached file system information was not valid.

GPFS_E_INVAL_IFILE
Incorrect `ifile` parameters.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_igetattrsx() subroutine

Retrieves extended file attributes; provides an option to include DMAPI attributes.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_igetattrsx(gpfs_ifile_t *ifile,
                    int flags,
                    void *buffer,
                    int bufferSize,
                    int *attrSize);
```

Description

The `gpfs_igetattrsx()` subroutine retrieves all extended file attributes in opaque format. It provides the same function as `gpfs_igetattrs()` but includes a `flags` parameter that allows the caller to back up and restore DMAPI attributes.

This function is intended for use by a backup program to save (and restore, using the related subroutine `gpfs_iputattrsx()`) all extended file attributes (ACLs, user attributes, and so forth) in one call. If the file does not have any extended attributes, the subroutine sets `attrSize` to zero.

Notes:

1. This call can optionally return extended attributes used for the Data Storage Management (XDSM) API (also known as DMAPI).
2. Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:
   - `libgpfs.a` for AIX
   - `libgpfs.so` for Linux

Parameters

ifile
Pointer to `gpfs_ifile_t` from `gpfs_iopen()`.

flags
Flags must have one of the following values:

GPFS_ATTRFLAG_NO_PLACEMENT
File attributes for placement are not saved, and neither is the current storage pool.

GPFS_ATTRFLAG_IGNORE_PLACEMENT
File attributes for placement are saved, but the current storage pool is not.

GPFS_ATTRFLAG_INCL_DMAPI
File attributes for DMAPI are included in the returned buffer.

GPFS_ATTRFLAG_USE_POLICY
Uses the restore policy rules to determine the pool ID.

GPFS_ATTRFLAG_INCL_DMAPI
Includes the DMAPI attributes.

GPFS_ATTRFLAG_FINALIZE_ATTRS
Finalizes immutability attributes.

GPFS_ATTRFLAG_SKIP_IMMUTABLE
Skips immutable attributes.

GPFS_ATTRFLAG_INCL_ENCR
Includes encryption attributes.

GPFS_ATTRFLAG_SKIP_CLONE
Skips clone attributes.

GPFS_ATTRFLAG_MODIFY_CLONEPARENT
Allows modification on the clone parent.

buffer
A pointer to the buffer for returned attributes.

bufferSize
Size of the buffer.

attrSize
Pointer to returned size of attributes.

Exit status

If the `gpfs_igetattrsx()` subroutine is successful, it returns a value of 0.

If the `gpfs_igetattrsx()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EINVAL
Not a GPFS file, or the flags provided are not valid.

ENOSPC
The buffer is too small to return all attributes. Field `attrSize` will be set to the size necessary.

ENOSYS
The `gpfs_igetattrsx()` subroutine is not available.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_igetfilesetname() subroutine

Returns the name of the fileset defined by a fileset ID.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_igetfilesetname(gpfs_iscan_t *iscan,
                         unsigned int filesetId,
                         void *buffer,
                         int bufferSize);
```

Description

The `gpfs_igetfilesetname()` subroutine is part of the backup by inode interface. The caller provides a pointer to the scan descriptor used to obtain the fileset ID. This library routine will return the name of the fileset defined by the fileset ID. The name is the null-terminated string provided by the administrator when the fileset was defined. The maximum string length is `GPFS_MAXNAMLEN`, which is defined in `/usr/lpp/mmfs/include/gpfs.h`.

Notes:

1. This routine is not thread safe. Only one thread at a time is allowed to invoke this routine for the given scan descriptor.
2. Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:
   - `libgpfs.a` for AIX
   - `libgpfs.so` for Linux

Parameters

iscan
Pointer to `gpfs_iscan_t` used to obtain the fileset ID.

filesetId
The fileset ID.

buffer
Pointer to buffer for returned attributes.

bufferSize
Size of the buffer.

Exit status

If the `gpfs_igetfilesetname()` subroutine is successful, it returns a value of 0.

If the `gpfs_igetfilesetname()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

E2BIG
The buffer is too small to return the fileset name.

EINTR
The call was interrupted. This routine is not thread safe.

EINVAL
The fileset ID is not valid.

ENOMEM
Unable to allocate memory for the request.

ENOSYS
The `gpfs_igetfilesetname()` subroutine is not available.

EPERM
The caller does not have superuser privileges.

ESTALE
The cached file system information was not valid.

GPFS_E_INVAL_ISCAN
The **iscan** parameters were not valid.

Examples

This programming segment gets the fileset name based on the given fileset ID. The returned fileset name is stored in `FileSetNameBuffer`, which has a length of `FileSetNameSize`.

```
gpfs_iscan_t *fsInodeScanP;
gpfs_igetfilesetname(fsInodeScanP,FileSetId, &FileSetNameBuffer,FileSetNameSize);
```

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_igetstoragepool() subroutine

Returns the name of the storage pool for the given storage pool ID.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_igetstoragepool(gpfs_iscan_t *iscan,
                         unsigned int dataPoolId,
                         void *buffer,
                         int bufferSize);
```

Description

The `gpfs_igetstoragepool()` subroutine is part of the backup by inode interface. The caller provides a pointer to the scan descriptor used to obtain the storage pool ID. This routine returns the name of the storage pool for the given storage pool ID. The name is the null-terminated string provided by the administrator when the storage pool was defined. The maximum string length is `GPFS_MAXNAMLEN`, which is defined in `/usr/lpp/mmfs/include/gpfs.h`.

Notes:

1. This routine is not thread safe. Only one thread at a time is allowed to invoke this routine for the given scan descriptor.
2. Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:
   - `libgpfs.a` for AIX
   - `libgpfs.so` for Linux

Parameters

iscan
Pointer to `gpfs_iscan_t` used to obtain the storage pool ID.

dataPoolId
The storage pool ID.

buffer
Pointer to buffer for returned attributes.

bufferSize
Size of the buffer.

Exit status

If the `gpfs_igetstoragepool()` subroutine is successful, it returns a value of 0.

If the `gpfs_igetstoragepool()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

E2BIG
The buffer is too small to return the storage pool name.

EINTR
The call was interrupted. This routine is not thread safe.

EINVAL
The storage pool ID is not valid.

ENOMEM
Unable to allocate memory for the request.

ENOSYS
The `gpfs_igetstoragepool()` subroutine is not available.

EPERM
The caller does not have superuser privileges.

ESTALE
The cached storage pool information was not valid.

GPFS_E_INVAL_ISCAN
The **iscan** parameters were not valid.

Examples

This programming segment gets the storage pool name based on the given storage pool ID. The returned storage pool name is stored in `StoragePoolNameBuffer` which has the length of `StoragePoolNameSize`.

```
gpfs_iscan_t *fsInodeScanP;
gpfs_igetstoragepool(fsInodeScanP,StgpoolIdBuffer, &StgpoolNameBuffer,StgpoolNameSize);
```

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_iopen() subroutine

Opens a file or directory by inode number.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
gpfs_ifile_t *gpfs_iopen(gpfs_fssnap_handle_t *fssnapHandle,
                         gpfs_ino_t ino,
                         int open_flags,
                         const gpfs_iattr_t *statxbuf,
                         const char *symLink);
```

Description

The `gpfs_iopen()` subroutine opens a user file or directory for backup. The file is identified by its inode number `ino` within the file system or snapshot identified by the `fssnapHandle`. The `fssnapHandle` parameter must be the same one that was used to create the inode scan that returned the inode number `ino`.

To read the file or directory, the `open_flags` must be set to GPFS_O_BACKUP. The `statxbuf` and `symLink` parameters are reserved for future use and must be set to NULL.

For an overview of using `gpfs_iopen()` in a backup application, see *Using APIs to develop backup applications* in *IBM Spectrum Scale: Administration Guide*.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fssnapHandle
File system snapshot handle.

ino
The inode number.

open_flags

GPFS_O_BACKUP
Read files for backup.

O_RDONLY
For `gpfs_iread()`.

statxbuf
This parameter is reserved for future use and should always be set to NULL.

symLink
This parameter is reserved for future use and should always be set to NULL.

Exit status

If the `gpfs_iopen()` subroutine is successful, it returns a pointer to the inode's file handle.

If the `gpfs_iopen()` subroutine is unsuccessful, it returns NULL and the global error variable `errno` is set to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EINVAL
Missing or incorrect parameter.

ENOENT
The file does not exist in the file system.

ENOMEM
Unable to allocate memory for the request.

ENOSYS
The `gpfs_iopen()` subroutine is not available.

EPERM
The caller does not have superuser privileges.

ESTALE
Cached file system information was not valid.

GPFS_E_INVAL_FSSNAPHANDLE
The file system snapshot handle is not valid.

GPFS_E_INVAL_INUM
Users are not authorized to open the reserved inodes.

Examples

For an example using `gpfs_iopen()`, see `/usr/lpp/mmfs/samples/util/tsreaddir.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_iopen64() subroutine

Opens a file or directory by inode number.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
gpfs_ifile_t *gpfs_iopen64(gpfs_fssnap_handle_t *fssnapHandle,
                           gpfs_ino64_t ino,
                           int open_flags,
                           const gpfs_iattr64_t *statxbuf,
                           const char *symLink);
```

Description

The `gpfs_iopen64()` subroutine opens a user file or directory for backup. The file is identified by its inode number `ino` within the file system or snapshot identified by the `fssnapHandle`. The `fssnapHandle` parameter must be the same one that was used to create the inode scan that returned the inode number `ino`.

To read the file or directory, the `open_flags` must be set to `GPFS_O_BACKUP`. The `statxbuf` and `symLink` parameters are reserved for future use and must be set to NULL.

For an overview of using `gpfs_iopen64()` in a backup application, see *Using APIs to develop backup applications* in *IBM Spectrum Scale: Administration Guide*.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fssnapHandle
The file system snapshot handle.

ino
The inode number.

open_flags

GPFS_O_BACKUP
Read files for backup.

O_RDONLY
For `gpfs_iread()`.

statxbuf
This parameter is reserved for future use and should always be set to NULL.

symLink
This parameter is reserved for future use and should always be set to NULL.

Exit status

If the `gpfs_iopen64()` subroutine is successful, it returns a pointer to the inode's file handle.

If the `gpfs_iopen64()` subroutine is unsuccessful, it returns NULL and the global error variable `errno` is set to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EFORMAT
The file system version number is not valid.

EINVAL
Missing or incorrect parameter.

ENOENT
The file does not exist in the file system.

ENOMEM
Unable to allocate memory for the request.

ENOSYS
The `gpfs_iopen64()` subroutine is not available.

EPERM
The caller does not have superuser privileges.

ESTALE
Cached file system information was not valid.

GPFS_E_INVAL_FSSNAPHANDLE
The file system snapshot handle is not valid.

GPFS_E_INVAL_IATTR
The `iattr` structure was corrupted.

GPFS_E_INVAL_INUM
Users are not authorized to open the reserved inodes.

Note: `gpfs_iopen64()` calls the standard library subroutines `dup()`, `open()`, and `malloc()`; if one of these called subroutines returns an error, `gpfs_iopen64()` also returns that error.

Examples

See the `gpfs_iopen()` example in `/usr/lpp/mmfs/samples/util/tsreaddir.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_iputattrsx() subroutine

Sets the extended file attributes for a file.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_iputattrsx(gpfs_ifile_t *ifile,
                    int flags,
                    void *buffer,
                    const char *pathName);
```

Description

The `gpfs_iputattrsx()` subroutine, together with `gpfs_igetattrsx()`, is intended for use by a backup program to save (`gpfs_igetattrsx()`) and restore (`gpfs_iputattrsx()`) all of the extended attributes of a file. This subroutine also sets the storage pool for the file and sets data replication to the values that are saved in the extended attributes.

This subroutine can optionally invoke the policy engine to match a RESTORE rule using the file's attributes saved in the extended attributes to set the file's storage pool and data replication as when calling `gpfs_fputattrswithpathname()`. When used with the policy engine, the caller should include the full path to the file, including the file name, to allow rule selection based on file name or path.

By default, the routine will not use RESTORE policy rules for data placement. The `pathName` parameter will be ignored and may be set to `NULL`.

If the call does not use RESTORE policy rules, or if the file fails to match a RESTORE rule, or if there are not RESTORE rules installed, then the storage pool and data replication are selected as when calling `gpfs_fputattrs()`.

The buffer passed in should contain extended attribute data that was obtained by a previous call to `gpfs_fgetattrs()`.

Note: This call will restore extended attributes used for the Data Storage Management (XDSM) API (also known as DMAPI) if they are present in the buffer.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

ifile
A pointer to `gpfs_ifile_t` from `gpfs_iopen()`.

flags
Flags must have one of the following values:

GPFS_ATTRFLAG_NO_PLACEMENT
File attributes are restored, but the storage pool and data replication are unchanged.

GPFS_ATTRFLAG_IGNORE_POOL
File attributes are restored, but the storage pool and data replication are selected by matching the saved attributes to a placement rule instead of restoring the saved storage pool.

GPFS_ATTRFLAG_USE_POLICY
File attributes are restored, but the storage pool and data replication are selected by matching the saved attributes to a RESTORE rule instead of restoring the saved storage pool.

GPFS_ATTRFLAG_USE_POLICY
Uses the restore policy rules to determine the pool ID.

GPFS_ATTRFLAG_INCL_DMAPI
Includes the DMAPI attributes.

GPFS_ATTRFLAG_FINALIZE_ATTRS
Finalizes immutability attributes.

GPFS_ATTRFLAG_SKIP_IMMUTABLE
Skips immutable attributes.

GPFS_ATTRFLAG_INCL_ENCR
Includes encryption attributes.

GPFS_ATTRFLAG_SKIP_CLONE
Skips clone attributes.

GPFS_ATTRFLAG_MODIFY_CLONEPARENT
Allows modification on the clone parent.

buffer
A pointer to the buffer containing the extended attributes for the file.

pathName
A pointer to a file path and file name. NULL is a valid value for `pathName`.

Note: `pathName` is a UTF-8 encoded string. On Windows, applications can convert UTF-16 (Unicode) to UTF-8 using the platform's `WideCharToMultiByte` function.

Exit status

If the `gpfs_iputattrsx()` subroutine is successful, it returns a value of 0.

If the `gpfs_iputattrsx()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EINVAL
The buffer pointed to by `buffer` does not contain valid attribute data, or invalid flags were provided.

ENOSYS
The `gpfs_iputattrsx()` subroutine is not supported under the current file system format.

EPERM
The caller of the subroutine must have superuser privilege.

ESTALE
The cached *fs* information was not valid.

GPFS_E_INVAL_IFILE
The `ifile` parameters provided were not valid.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

/usr/lpp/mmfs/lib/libgpfs.so for Linux

gpfs_iread() subroutine

Reads a file opened by gpfs_iopen().

Library

GPFS Library (libgpfs.a for AIX, libgpfs.so for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_iread(gpfs_ifile_t *ifile,
               void *buffer,
               int bufferSize,
               gpfs_off64_t *offset);
```

Description

The gpfs_iread() subroutine reads data from the file indicated by the ifile parameter returned from gpfs_iopen(). This subroutine reads data beginning at parameter offset and continuing for bufferSize bytes into the buffer specified by buffer. If successful, the subroutine returns a value that is the length of the data read, and sets parameter offset to the offset of the next byte to be read. A return value of 0 indicates end-of-file.

For an overview of using gpfs_iread() in a backup application, see *Using APIs to develop backup applications* in *IBM Spectrum Scale: Administration Guide*.

Note: Compile any program that uses this subroutine with the -lgpfs flag from the following library:

- libgpfs.a for AIX
- libgpfs.so for Linux

Parameters

ifile
Pointer to gpfs_ifile_t from gpfs_iopen().

buffer
Buffer for the data to be read.

bufferSize
Size of the buffer (that is, the amount of data to be read).

offset
Offset of where within the file to read. If gpfs_iread() is successful, offset is updated to the next byte after the last one that was read.

Exit status

If the gpfs_iread() subroutine is successful, it returns the number of bytes read.

If the gpfs_iread() subroutine is unsuccessful, it returns a value of -1 and sets the global error variable errno to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EISDIR
The specified file is a directory.

EINVAL
Missing or incorrect parameter.

ENOSYS
The `gpfs_iread()` subroutine is not available.

EPERM
The caller does not have superuser privileges.

ESTALE
Cached file system information was not valid.

GPFS_E_INVAL_IFILE
Incorrect `ifile` parameter.

GPFS_E_ISLNK
The specified file is a symlink. Use `gpfs_ireadlink` subroutine on symlink.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_ireaddir() subroutine

Reads the next directory entry.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_ireaddir(gpfs_ifile_t *idir,
                  const gpfs_direntx_t **dirent);
```

Description

The `gpfs_ireaddir()` subroutine returns the next directory entry in a file system. For an overview of using `gpfs_ireaddir()` in a backup application, see *Using APIs to develop backup applications* in *IBM Spectrum Scale: Administration Guide*.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

idir
Pointer to `gpfs_ifile_t` from `gpfs_iopen()`.

dirent
Pointer to returned pointer to directory entry.

Exit status

If the `gpfs_ireaddir()` subroutine is successful, it returns a value of 0 and sets the `dirent` parameter to point to the returned directory entry. If there are no more GPFS directory entries, `gpfs_ireaddir()` returns a value of 0 and sets the `dirent` parameter to NULL.

If the `gpfs_ireaddir()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

ENOMEM
Unable to allocate memory for the request.

ENOSYS
The `gpfs_ireaddir()` subroutine is not available.

ENOTDIR
File is not a directory.

EPERM
The caller does not have superuser privileges.

ESTALE
The cached file system information was not valid.

GPFS_E_INVAL_IFILE
Incorrect `ifile` parameter.

Examples

For an example using `gpfs_ireaddir()`, see `/usr/lpp/mmfs/samples/util/tsreaddir.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_ireaddir64() subroutine

Reads the next directory entry.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_ireaddir64(gpfs_ifile_t *idir,
                    const gpfs_direntx64_t **dirent);
```

Description

The `gpfs_ireaddir64()` subroutine returns the next directory entry in a file system. For an overview of using `gpfs_ireaddir64()` in a backup application, see *Using APIs to develop backup applications* in *IBM Spectrum Scale: Administration Guide*.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

idir
A pointer to `gpfs_ifile_t` from `gpfs_iopen64()`.

dirent
A pointer to the returned pointer to the directory entry.

Exit status

If the `gpfs_ireaddir64()` subroutine is successful, it returns a value of 0 and sets the `dirent` parameter to point to the returned directory entry. If there are no more GPFS directory entries, `gpfs_ireaddir64()` returns a value of 0 and sets the `dirent` parameter to NULL.

If the `gpfs_ireaddir64()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

ENOMEM
Unable to allocate memory for the request.

ENOSYS
The `gpfs_ireaddir64()` subroutine is not available.

ENOTDIR
File is not a directory.

EPERM
The caller does not have superuser privileges.

ESTALE
The cached file system information was not valid.

GPFS_E_INVAL_IFILE
Incorrect `ifile` parameter.

Examples

See the `gpfs_ireaddir()` example in `/usr/lpp/mmfs/samples/util/tsreaddir.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_ireadlink() subroutine

Reads a symbolic link by inode number.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_ireadlink(gpfs_fssnap_handle_t *fssnapHandle,
                   gpfs_ino_t ino,
                   char *buffer,
                   int bufferSize);
```

Description

The `gpfs_ireadlink()` subroutine reads a symbolic link by inode number. Like `gpfs_iopen()`, it uses the same `fssnapHandle` parameter that was used by the inode scan.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fssnapHandle
File system snapshot handle.

ino
inode number of the link file to read.

buffer
Pointer to buffer for the returned link data.

bufferSize
Size of the buffer.

Exit status

If the `gpfs_ireadlink()` subroutine is successful, it returns the number of bytes read.

If the `gpfs_ireadlink()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EINVAL
Missing or incorrect parameter.

ENOENT
No such file or directory.

ENOMEM
Unable to allocate memory for the request.

ENOSYS
The `gpfs_ireadlink()` subroutine is not available.

EPERM
The caller does not have superuser privileges.

ERANGE
On AIX, the buffer is too small to return the symbolic link.

ESTALE
Cached file system information was not valid.

GPFS_E_INVAL_FSSNAPHANDLE
The file system snapshot handle is not valid.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_ireadlink64() subroutine

Reads a symbolic link by inode number.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_ireadlink64(gpfs_fssnap_handle_t *fssnapHandle,
                     gpfs_ino64_t ino,
                     char *buffer,
                     int bufferSize);
```

Description

The `gpfs_ireadlink64()` subroutine reads a symbolic link by inode number. Like `gpfs_iopen64()`, it uses the same `fssnapHandle` parameter that was used by the inode scan.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fssnapHandle
The file system snapshot handle.

ino
The inode number of the link file to read.

buffer
A pointer to buffer for the returned link data.

bufferSize
The size of the buffer.

Exit status

If the `gpfs_ireadlink64()` subroutine is successful, it returns the number of bytes read.

If the `gpfs_ireadlink64()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EINVAL
Missing or incorrect parameter.

ENOENT
No such file or directory.

ENOMEM
Unable to allocate memory for the request.

ENOSYS
The `gpfs_ireadlink64()` subroutine is not available.

EPERM
The caller does not have superuser privileges.

ERANGE
On AIX, the buffer is too small to return the symbolic link.

ESTALE
The cached file system information was not valid.

GPFS_E_INVAL_FSSNAPHANDLE
The file system snapshot handle is not valid.

Note: `gpfs_ireadlink64()` calls the standard library subroutine `readlink()`; if this called subroutine returns an error, `gpfs_ireadlink64()` also returns that error.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_ireadx() subroutine

Performs block level incremental read of a file within an incremental inode scan.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
gpfs_off64_t gpfs_ireadx(gpfs_ifile_t *ifile,
                         gpfs_iscan_t *iscan,
                         void *buffer,
                         int bufferSize,
                         gpfs_off64_t *offset,
                         gpfs_off64_t termOffset,
                         int *hole);
```

Description

The `gpfs_ireadx()` subroutine performs a block level incremental read on a file opened by `gpfs_iopen()` within a given incremental scan opened using `gpfs_open_inodescan()`.

For an overview of using `gpfs_ireadx()` in a backup application, see *Using APIs to develop backup applications* in *IBM Spectrum Scale: Administration Guide*.

The `gpfs_ireadx()` subroutine returns the data that has changed since the `prev_fssnapId` specified for the inode scan. The file is scanned starting at `offset` and terminating at `termOffset`, looking for changed data. Once changed data is located, the `offset` parameter is set to its location, the new data is returned in the `buffer` provided, and the amount of data returned is the subroutine's value.

If the change to the data is that it has been deleted (that is, the file has been truncated), no data is returned, but the `hole` parameter is returned with a value of `1`, and the size of the `hole` is returned as the subroutine's value. The returned size of the hole may exceed the `bufferSize` provided. If no changed data was found before reaching the `termOffset` or the end-of-file, then the `gpfs_ireadx()` subroutine return value is 0.

Block level incremental backups are not available on small files (a file size smaller than the file system block size), directories, or if the file has been deleted. The `gpfs_ireadx()` subroutine can still be used, but it returns all of the file's data, operating like the standard `gpfs_iread()` subroutine. However, the `gpfs_ireadx()` subroutine will still identify sparse files and explicitly return information on holes in the files, rather than returning the NULL data.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

ifile
Pointer to `gpfs_ifile_t` returned from `gpfs_iopen()`.

iscan
Pointer to `gpfs_iscan_t` from `gpfs_open_inodescan()`.

buffer
Pointer to buffer for returned data, or NULL to query the next increment to be read.

bufferSize
Size of buffer for returned data.

offset
On input, the offset to start the scan for changes. On output, the offset of the changed data, if any was detected.

termOffset
Read terminates before reading this offset. The caller may specify `ia_size` from the file's `gpfs_iattr_t` or 0 to scan the entire file.

hole
Pointer to a flag returned to indicate a hole in the file. A value of 0 indicates that the `gpfs_ireadx()` subroutine returned data in the `buffer`. A value of 1 indicates that `gpfs_ireadx()` encountered a hole at the returned `offset`.

Exit status

If the `gpfs_ireadx()` subroutine is successful, it returns the number of bytes read and returned in `bufP`, or the size of the hole encountered in the file.

If the `gpfs_ireadx()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EDOM
The file system stripe ID from the `iscanId` does not match the `ifile`'s.

EINVAL
Missing or incorrect parameter.

EISDIR
The specified file is a directory.

ENOMEM
Unable to allocate memory for the request.

ENOSYS
The `gpfs_ireadx()` subroutine is not available.

EPERM
The caller does not have superuser privileges.

ERANGE
The file system snapshot ID from the `iscanId` is more recent than the `ifile`'s.

ESTALE
Cached file system information was not valid.

GPFS_E_INVAL_IFILE
Incorrect `ifile` parameter.

GPFS_E_INVAL_ISCAN
Incorrect `iscan` parameter.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_iscan_t structure

Contains a handle for an inode scan of a GPFS file system or snapshot.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Structure

```
typedef struct gpfs_iscan gpfs_iscan_t;
```

Description

The `gpfs_iscan_t` structure contains a handle for an inode scan of a GPFS file system or snapshot.

Members

gpfs_iscan
The handle for an inode scan for a GPFS file system or snapshot.

Examples

For an example using `gpfs_iscan_t`, see `/usr/lpp/mmfs/samples/util/tstimes.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_lib_init() subroutine

Sets up a GPFS interface for additional calls.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_lib_init(int flags);
```

Description

The `gpfs_lib_init()` subroutine, together with the `gpfs_lib_term()` subroutine, is intended for use by a program that makes repeated calls to a GPFS programming interface. This subroutine sets up the internal structure to speed up additional interface calls.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

flags
Reserved for future use. Must be zero.

Exit status

If the `gpfs_lib_init()` subroutine is successful, it returns a value of 0.

If the `gpfs_lib_init()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EINVAL
A nonzero value was passed as the `flags` parameter.

ENOSYS
The `gpfs_lib_init()` subroutine is not supported under the current file system format.

Examples

For an example using `gpfs_lib_init()`, see `/usr/lpp/mmfs/samples/util/tsfindinode.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_lib_term() subroutine

Cleans up after GPFS interface calls have been completed.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_lib_term(int flags);
```

Description

The `gpfs_lib_term()` subroutine, together with the `gpfs_lib_init()` subroutine, is intended for use by a program that makes repeated calls to a GPFS programming interface. This subroutine cleans up the internal structure previously set up by `gpfs_lib_init()`.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

flags
Reserved for future use. Must be zero.

Exit status

If the `gpfs_lib_term()` subroutine is successful, it returns a value of 0.

If the `gpfs_lib_term()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EINTR
The `gpfs_lib_term()` subroutine was interrupted by a signal that was caught. Cleanup was done.

EINVAL
A nonzero value was passed as the `flags` parameter.

Examples

For an example using `gpfs_lib_term()`, see `/usr/lpp/mmfs/samples/util/tsfindinode.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_next_inode() subroutine

Retrieves the next inode from the inode scan.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_next_inode(gpfs_iscan_t *iscan,
                    gpfs_ino_t termIno,
                    const gpfs_iattr_t **iattr);
```

Description

The `gpfs_next_inode()` subroutine obtains the next inode from the specified inode scan and sets the `iattr` pointer to the inode's attributes. The `termIno` parameter can be used to terminate the inode scan before the last inode in the file system or snapshot being scanned. A value of 0 may be provided to indicate the last inode in the file system or snapshot. If there are no more inodes to be returned before the termination inode, the `gpfs_next_inode()` subroutine returns a value of 0 and the inode's attribute pointer is set to NULL.

For an overview of using `gpfs_next_inode()` in a backup application, see *Using APIs to develop backup applications* in *IBM Spectrum Scale: Administration Guide*.

To generate a full backup, invoke `gpfs_open_inodescan()` with NULL for the `prev_fssnapId` parameter. Repeated invocations of `gpfs_next_inode()` then return inode information about all existing user files, directories, and links in inode number order.

To generate an incremental backup, invoke `gpfs_next_inode()` with the `fssnapId` that was obtained from a `fssnapHandle` at the time the previous backup was created. The snapshot that was used for the previous backup does not need to exist at the time the incremental backup is generated. That is, the backup application needs to remember only the `fssnapId` of the previous backup; the snapshot itself can be deleted as soon as the backup is completed.

For an incremental backup, only inodes of files that have changed since the specified previous snapshot will be returned. Any operation that changes the file's `mtime` or `ctime` is considered a change and will cause the file to be included. Files with no changes to the file's data or file attributes, other than a change to `atime`, are omitted from the scan.

Incremental backups return deleted files, but full backups do not. A deleted file is indicated by the field `ia_nlinks` having a value of 0.

To read only the inodes that have been copied to a snapshot, use `gpfs_open_inodescan()` with `fssnapHandle` of the snapshot and pass `fssnapid` of the `fssnapHandle` as `prev_fssnapId`. Repeated invocations of `gpfs_next_inode()` then return the inodes copied to the snapshot, skipping holes.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

iscan
Pointer to the inode scan handle.

termIno
The inode scan terminates before this inode number. The caller may specify `maxIno` from `gpfs_open_inodescan()` or zero to scan the entire inode file.

iattr
Pointer to the returned pointer to the inode's `iattr`.

Exit status

If the `gpfs_next_inode()` subroutine is successful, it returns a value of 0 and a pointer. The pointer points to NULL if there are no more inodes. Otherwise, the pointer points to the returned inode's attributes.

If the `gpfs_next_inode()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

ENOMEM
Unable to allocate memory for the request.

ENOSYS
The `gpfs_next_inode()` subroutine is not available.

EPERM
The caller does not have superuser privileges.

ESTALE
Cached file system information was not valid.

GPFS_E_INVAL_FSSNAPID
The file system snapshot ID is not valid.

GPFS_E_INVAL_ISCAN
Incorrect parameters.

Examples

For an example using `gpfs_next_inode()`, see `/usr/lpp/mmfs/samples/util/tstimes.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_next_inode64() subroutine

Retrieves the next inode from the inode scan.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_next_inode64(gpfs_iscan_t *iscan,
                      gpfs_ino64_t termIno,
                      const gpfs_iattr64_t **iattr);
```

Description

The `gpfs_next_inode64()` subroutine obtains the next inode from the specified inode scan and sets the `iattr` pointer to the inode's attributes. The `termIno` parameter can be used to stop the inode scan before the last inode in the file system or snapshot being scanned. A value of 0 can be provided to indicate the last inode in the file system or snapshot. If there are no more inodes to be returned before the termination inode, the `gpfs_next_inode64()` subroutine returns a value of 0 and the inode's attribute pointer is set to NULL.

For an overview of using `gpfs_next_inode64()` in a backup application, see *Using APIs to develop backup applications* in *IBM Spectrum Scale: Administration Guide*.

To generate a full backup, invoke `gpfs_open_inodescan64()` with NULL for the `prev_fssnapId` parameter. Repeated invocations of `gpfs_next_inode64()` then return inode information about all existing user files, directories, and links in inode number order.

To generate an incremental backup, invoke `gpfs_next_inode64()` with the `fssnapId` that was obtained from a `fssnapHandle` at the time the previous backup was created. The snapshot that was used for the previous backup does not need to exist at the time the incremental backup is generated. That is, the backup application needs to remember only the `fssnapId` of the previous backup; the snapshot itself can be deleted as soon as the backup is completed.

For an incremental backup, only inodes of files that have changed since the specified previous snapshot will be returned. Any operation that changes the file's `mtime` or `ctime` is considered a change and will cause the file to be included. Files with no changes to the file's data or file attributes, other than a change to `atime`, are omitted from the scan.

Incremental backups return deleted files, but full backups do not. A deleted file is indicated by the field `ia_nlinks` having a value of 0.

To read only the inodes that have been copied to a snapshot, use `gpfs_open_inodescan64()` with `fssnapHandle` of the snapshot and pass `fssnapid` of the `fssnapHandle` as `prev_fssnapId`. Repeated invocations of `gpfs_next_inode64()` then return the inodes copied to the snapshot, skipping holes.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

iscan
A pointer to the inode scan handle.

termIno
The inode scan terminates before this inode number. The caller may specify `maxIno` from `gpfs_open_inodescan64()` or zero to scan the entire inode file.

iattr
A pointer to the returned pointer to the inode's `iattr`.

Exit status

If the `gpfs_next_inode64()` subroutine is successful, it returns a value of 0 and a pointer. The pointer points to NULL if there are no more inodes. Otherwise, the pointer points to the returned inode's attributes.

If the `gpfs_next_inode64()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

ENOMEM
Unable to allocate memory for the request.

ENOSYS
The `gpfs_next_inode64()` subroutine is not available.

EPERM
The caller does not have superuser privileges.

ESTALE
The cached file system information was not valid.

GPFS_E_INVAL_FSSNAPID
The file system snapshot ID is not valid.

GPFS_E_INVAL_ISCAN
Incorrect parameters.

Examples

See the `gpfs_next_inode()` example in `/usr/lpp/mmfs/samples/util/tstimes.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_next_inode_with_xattrs() subroutine

Retrieves the next inode and its extended attributes from the inode scan.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_next_inode_with_xattrs(gpfs_iscan_t *iscan,
                                gpfs_ino_t termIno,
                                const gpfs_iattr_t **iattr,
                                const char **xattrBuf,
                                unsigned int *xattrBufLen);
```

Description

The `gpfs_next_inode_with_xattrs()` subroutine retrieves the next inode and its extended attributes from the inode scan. The set of extended attributes returned are defined when the inode scan was opened. The scan stops before the last inode that was specified or the last inode in the inode file being scanned.

The data returned by `gpfs_next_inode()` is overwritten by subsequent calls to `gpfs_next_inode()`, `gpfs_seek_inode()`, or `gpfs_stat_inode()`.

The `termIno` parameter provides a way to partition an inode scan so it can be run on more than one node.

The returned values for `xattrBuf` and `xattrBufLen` must be provided to `gpfs_next_xattr()` to obtain the extended attribute names and values. The buffer used for the extended attributes is overwritten by subsequent calls to `gpfs_next_inode()`, `gpfs_seek_inode()`, or `gpfs_stat_inode()`.

The returned pointers to the extended attribute name and value will be aligned to a double-word boundary.

Parameters

iscan
A pointer to the inode scan descriptor.

termIno
The inode scan stops before this inode number. The caller can specify `maxIno` from `gpfs_open_inodescan()` or zero to scan the entire inode file.

iattr
A pointer to the returned pointer to the file's `iattr`.

xattrBuf
A pointer to the returned pointer to the `xiattr` buffer.

xattrBufLen
The returned length of the `xiattr` buffer.

Exit status

If the `gpfs_next_inode_with_xattrs()` subroutine is successful, it returns a value of 0 and `iattr` is set to point to `gpfs_iattr_t`. The pointer points to NULL if there are no more inodes, otherwise, the pointer points to `gpfs_iattr_t`.

If the gpfs_next_inode_with_xattrs() subroutine is unsuccessful, it returns a value of -1 and sets the global error variable errno to NULL to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EFAULT
The buffer data was overwritten.

ENOMEM
The buffer is too small, unable to allocate memory for the request.

ENOSYS
The gpfs_next_inode_with_xattrs() subroutine is not available.

EPERM
The caller does not have superuser privileges.

ESTALE
The cached file system information was not valid.

GPFS_E_INVAL_ISCAN
Incorrect parameters.

GPFS_E_INVAL_XATTR
Incorrect parameters.

Examples

For an example using gpfs_next_inode_with_xattrs(), see /usr/lpp/mmfs/samples/util/tsinode.c.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX

/usr/lpp/mmfs/lib/libgpfs.so for Linux

gpfs_next_inode_with_xattrs64() subroutine

Retrieves the next inode and its extended attributes from the inode scan.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_next_inode_with_xattrs64(gpfs_iscan_t *iscan,
                                  gpfs_ino64_t termIno,
                                  const gpfs_iattr64_t **iattr,
                                  const char **xattrBuf,
                                  unsigned int *xattrBufLen);
```

Description

The `gpfs_next_inode_with_xattrs64()` subroutine retrieves the next inode and its extended attributes from the inode scan. The set of extended attributes returned are defined when the inode scan was opened. The scan stops before the last inode that was specified or the last inode in the inode file being scanned.

The data returned by `gpfs_next_inode64()` is overwritten by subsequent calls to `gpfs_next_inode64()`, `gpfs_seek_inode64()`, or `gpfs_stat_inode64()`.

The `termIno` parameter provides a way to partition an inode scan so it can be run on more than one node.

The returned values for `xattrBuf` and `xattrBufLen` must be provided to `gpfs_next_xattr()` to obtain the extended attribute names and values. The buffer used for the extended attributes is overwritten by subsequent calls to `gpfs_next_inode64()`, `gpfs_seek_inode64()`, or `gpfs_stat_inode64()`.

The returned pointers to the extended attribute name and value will be aligned to a double-word boundary.

Parameters

iscan
A pointer to the inode scan descriptor.

termIno
The inode scan stops before this inode number. The caller can specify `maxIno` from `gpfs_open_inodescan64()` or zero to scan the entire inode file.

iattr
A pointer to the returned pointer to the file's `iattr`.

xattrBuf
A pointer to the returned pointer to the `xiattr` buffer. Initialize this parameter to a valid value or NULL before calling `gpfs_next_inode_with_xattrs64`.

xattrBufLen
The returned length of the `xiattr` buffer. Initialize this parameter to a valid value or NULL before calling `gpfs_next_inode_with_xattrs64`.

Exit status

If the gpfs_next_inode_with_xattrs64() subroutine is successful, it returns a value of 0 and iattr is set to point to gpfs_iattr_t. The pointer points to NULL if there are no more inodes, otherwise, the pointer points to gpfs_iattr_t.

If the gpfs_next_inode_with_xattrs64() subroutine is unsuccessful, it returns a value of -1 and sets the global error variable errno to NULL to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EFAULT
The buffer data was overwritten.

ENOMEM
Unable to allocate memory for the request.

ENOSYS
The gpfs_next_inode_with_xattrs64() subroutine is not available.

EPERM
The caller does not have superuser privileges.

ESTALE
The cached file system information was not valid.

GPFS_E_INVAL_ISCAN
Incorrect parameters.

GPFS_E_INVAL_XATTR
Incorrect parameters.

Examples

See the gpfs_next_inode_with_xattrs() example in /usr/lpp/mmfs/samples/util/tsinode.c.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX

/usr/lpp/mmfs/lib/libgpfs.so for Linux

gpfs_next_xattr() subroutine

Returns individual attributes and their values.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_next_xattr(gpfs_iscan_t *iscan,
                    const char **xattrBuf,
                    unsigned int *xattrBufLen,
                    const char **name,
                    unsigned int *valueLen,
                    const char **value);
```

Description

The `gpfs_next_xattr()` subroutine iterates over the extended attributes buffer returned by the `gpfs_next_inode_with_xattrs()` or `gpfs_next_inode_with_xattrs64()` subroutine to return the individual attributes and their values. The attribute names are null-terminated strings, whereas the attribute value contains binary data.

Note: The caller is not allowed to modify the returned attribute names or values. The data returned by `gpfs_next_xattr()` might be overwritten by subsequent calls to `gpfs_next_xattr()` or other GPFS library calls.

Parameters

iscan
A pointer to the inode descriptor.

xattrBuf
A pointer to the pointer to the attribute buffer.

xattrBufLen
A pointer to the attribute buffer length.

name
A pointer to the attribute name.

valueLen
A pointer to the length of the attribute value.

value
A pointer to the attribute value.

Exit status

If the `gpfs_next_xattr()` subroutine is successful, it returns a value of 0 and a pointer to the attribute name. It also sets:

- The `valueLen` parameter to the length of the attribute value
- The `value` parameter to point to the attribute value
- The `xattrBufLen` parameter to the remaining length of buffer
- The `xattrBuf` parameter to index the next attribute in buffer

If the `gpfs_next_xattr()` subroutine is successful, but there are no more attributes in the buffer, it returns a value of 0 and the attribute name is set to NULL. It also sets:

- The valueLen parameter to 0
- The value parameter to NULL
- The xattrBufLen parameter to 0
- The xattrBuf parameter to NULL

If the gpfs_next_xattr() subroutine is unsuccessful, it returns a value of -1 and sets the global error variable errno to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EINVAL
Incorrect parameters.

ENOSYS
The gpfs_next_xattr() subroutine is not available.

Examples

For an example using gpfs_next_xattr(), see /usr/lpp/mmfs/samples/util/tsinode.c.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX

/usr/lpp/mmfs/lib/libgpfs.so for Linux

gpfs_opaque_acl_t structure

Contains buffer mapping for the gpfs_getacl() and gpfs_putacl() subroutines.

Library

GPFS Library (libgpfs.a for AIX, libgpfs.so for Linux)

Structure

```
typedef struct
{
  int             acl_buffer_len;
  unsigned short  acl_version;
  unsigned char   acl_type;
  char            acl_var_data[1];
} gpfs_opaque_acl_t;
```

Description

The gpfs_opaque_acl_t structure contains size, version, and ACL type information for the gpfs_getacl() and gpfs_putacl() subroutines.

Members

acl_buffer_len
On input, this field must be set to the total length, in bytes, of the data structure being passed to GPFS. On output, this field contains the actual size of the requested information. If the initial size of the buffer is not large enough to contain all of the information, the gpfs_getacl() invocation must be repeated with a larger buffer.

acl_version
This field contains the current version of the GPFS internal representation of the ACL. On input to the gpfs_getacl() subroutine, set this field to zero.

acl_type
On input to the gpfs_getacl() subroutine, set this field to either GPFS_ACL_TYPE_ACCESS or GPFS_ACL_TYPE_DEFAULT, depending on which ACL is requested. These constants are defined in the gpfs.h header file.

acl_var_data
This field signifies the beginning of the remainder of the ACL information.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX

/usr/lpp/mmfs/lib/libgpfs.so for Linux

gpfs_open_inodescan() subroutine

Opens an inode scan of a file system or snapshot.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
gpfs_iscan_t *gpfs_open_inodescan(gpfs_fssnap_handle_t *fssnapHandle,
                                  const gpfs_fssnap_id_t *prev_fssnapId,
                                  gpfs_ino_t *maxIno);
```

Description

The `gpfs_open_inodescan()` subroutine opens a scan of the inodes in the file system or snapshot identified by the `fssnapHandle` parameter. The scan traverses all user files, directories and links in the file system or snapshot. The scan begins with the user file with the lowest inode number and returns the files in increasing order. The `gpfs_seek_inode()` subroutine may be used to set the scan position to an arbitrary inode. System files, such as the block allocation maps, are omitted from the scan. The file system must be mounted to open an inode scan.

For an overview of using `gpfs_open_inodescan()` in a backup application, see *Using APIs to develop backup applications* in *IBM Spectrum Scale: Administration Guide*.

To generate a full backup, invoke `gpfs_open_inodescan()` with NULL for the `prev_fssnapId` parameter. Repeated invocations of `gpfs_next_inode()` then return inode information about all existing user files, directories, and links in inode number order.

To generate an incremental backup, invoke `gpfs_open_inodescan()` with the `fssnapId` that was obtained from a `fssnapHandle` at the time the previous backup was created. The snapshot that was used for the previous backup does not need to exist at the time the incremental backup is generated. That is, the backup application needs to remember only the `fssnapId` of the previous backup; the snapshot itself can be deleted as soon as the backup is completed.

For the incremental backup, any operation that changes the file's `mtime` or `ctime` causes the file to be included. Files with no changes to the file's data or file attributes, other than a change to `atime`, are omitted from the scan.

A full inode scan (`prev_fssnapId` set to NULL) does not return any inodes of nonexistent or deleted files, but an incremental inode scan (`prev_fssnapId` not NULL) does return inodes for files that have been deleted since the previous snapshot. The inodes of deleted files have a link count of zero.

If the snapshot indicated by `prev_fssnapId` is available, the caller may benefit from the extended read subroutine, `gpfs_ireadx()`, which returns only the changed blocks within the files. Without the previous snapshot, all blocks within the changed files are returned.

Once a full or incremental backup completes, the `new_fssnapId` must be saved in order to reuse it on a subsequent incremental backup. This `fssnapId` must be provided to the `gpfs_open_inodescan()` subroutine, as the `prev_fssnapId` input parameter.

To read only the inodes that have been copied to a snapshot, use `gpfs_open_inodescan()` with `fssnapHandle` of the snapshot and pass `fssnapid` of the `fssnapHandle` as the `prev_fssnapId`.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fssnapHandle
File system snapshot handle.

prev_fssnapId
Pointer to file system snapshot ID or NULL. If `prev_fssnapId` is provided, the inode scan returns only the files that have changed since the previous backup. If the pointer is NULL, the inode scan returns all user files.

If it is the same as the fssnapid of the fssnapHandle parameter, the scan only returns the inodes copied into the corresponding snapshot.

maxIno
Pointer to inode number or NULL. If provided, `gpfs_open_inodescan()` returns the maximum inode number in the file system or snapshot being scanned.

Exit status

If the `gpfs_open_inodescan()` subroutine is successful, it returns a pointer to an inode scan handle.

If the `gpfs_open_inodescan()` subroutine is unsuccessful, it returns a NULL pointer and the global error variable `errno` is set to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EDOM
The file system snapshot ID passed for `prev_fssnapId` is from a different file system.

EINVAL
Incorrect parameters.

ENOMEM
Unable to allocate memory for the request.

ENOSYS
The `gpfs_open_inodescan()` subroutine is not available.

EPERM
The caller does not have superuser privileges.

ERANGE
The `prev_fssnapId` parameter is the same as or more recent than `snapId` being scanned.

ESTALE
Cached file system information was not valid.

GPFS_E_INVAL_FSSNAPHANDLE
The file system snapshot handle is not valid.

GPFS_E_INVAL_FSSNAPID
The file system snapshot ID passed for `prev_fssnapId` is not valid.

Examples

For an example using `gpfs_open_inodescan()`, see `/usr/lpp/mmfs/samples/util/tstimes.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_open_inodescan64() subroutine

Opens an inode scan of a file system or snapshot.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
gpfs_iscan_t *gpfs_open_inodescan64(gpfs_fssnap_handle_t *fssnapHandle,
                                    const gpfs_fssnap_id_t *prev_fssnapId,
                                    gpfs_ino64_t *maxIno);
```

Description

The `gpfs_open_inodescan64()` subroutine opens a scan of the inodes in the file system or snapshot identified by the `fssnapHandle` parameter. The scan traverses all user files, directories and links in the file system or snapshot. The scan begins with the user file with the lowest inode number and returns the files in increasing order. The `gpfs_seek_inode64()` subroutine may be used to set the scan position to an arbitrary inode. System files, such as the block allocation maps, are omitted from the scan. The file system must be mounted to open an inode scan.

For an overview of using `gpfs_open_inodescan64()` in a backup application, see *Using APIs to develop backup applications* in *IBM Spectrum Scale: Administration Guide*.

To generate a full backup, invoke `gpfs_open_inodescan64()` with NULL for the `prev_fssnapId` parameter. Repeated invocations of `gpfs_next_inode64()` then return inode information about all existing user files, directories, and links in inode number order.

To generate an incremental backup, invoke `gpfs_open_inodescan64()` with the `fssnapId` that was obtained from a `fssnapHandle` at the time the previous backup was created. The snapshot that was used for the previous backup does not need to exist at the time the incremental backup is generated. That is, the backup application needs to remember only the `fssnapId` of the previous backup; the snapshot itself can be deleted as soon as the backup is completed.

For the incremental backup, any operation that changes the file's `mtime` or `ctime` causes the file to be included. Files with no changes to the file's data or file attributes, other than a change to `atime`, are omitted from the scan.

A full inode scan (`prev_fssnapId` set to NULL) does not return any inodes of nonexistent or deleted files, but an incremental inode scan (`prev_fssnapId` not NULL) does return inodes for files that have been deleted since the previous snapshot. The inodes of deleted files have a link count of zero.

If the snapshot indicated by `prev_fssnapId` is available, the caller may benefit from the extended read subroutine, `gpfs_ireadx()`, which returns only the changed blocks within the files. Without the previous snapshot, all blocks within the changed files are returned.

Once a full or incremental backup completes, the `new_fssnapId` must be saved in order to reuse it on a subsequent incremental backup. This `fssnapId` must be provided to the `gpfs_open_inodescan64()` subroutine, as the `prev_fssnapId` input parameter.

To read only the inodes that have been copied to a snapshot, use `gpfs_open_inodescan()` with `fssnapHandle` of the snapshot and pass `fssnapid` of the `fssnapHandle` as the `prev_fssnapId`.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fssnapHandle
The file system snapshot handle.

prev_fssnapId
A pointer to file system snapshot ID or NULL. If `prev_fssnapId` is provided, the inode scan returns only the files that have changed since the previous backup. If the pointer is NULL, the inode scan returns all user files.

If it is same as the fssnapid of the fssnapHandle parameter, the scan only returns the inodes copied into the corresponding snapshot.

maxIno
A pointer to inode number or NULL. If provided, `gpfs_open_inodescan64()` returns the maximum inode number in the file system or snapshot being scanned.

Exit status

If the `gpfs_open_inodescan64()` subroutine is successful, it returns a pointer to an inode scan handle.

If the `gpfs_open_inodescan64()` subroutine is unsuccessful, it returns a NULL pointer and the global error variable `errno` is set to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EDOM
The file system snapshot ID passed for `prev_fssnapId` is from a different file system.

EINVAL
Incorrect parameters.

ENOMEM
Unable to allocate memory for the request.

ENOSYS
The `gpfs_open_inodescan64()` subroutine is not available.

EPERM
The caller does not have superuser privileges.

ERANGE
The `prev_fssnapId` parameter is the same as or more recent than `snapId` being scanned.

ESTALE
The cached file system information was not valid.

GPFS_E_INVAL_FSSNAPHANDLE
The file system snapshot handle is not valid.

GPFS_E_INVAL_FSSNAPID
The file system snapshot ID passed for `prev_fssnapId` is not valid.

Note: `gpfs_open_inodescan64()` calls the standard library subroutines `dup()` and `malloc()`; if one of these called subroutines returns an error, `gpfs_open_inodescan64()` also returns that error.

Examples

See the `gpfs_open_inodescan()` example in `/usr/lpp/mmfs/samples/util/tstimes.c`.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX

/usr/lpp/mmfs/lib/libgpfs.so for Linux

gpfs_open_inodescan_with_xattrs() subroutine

Opens an inode file and extended attributes for an inode scan.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
gpfs_iscan_t *gpfs_open_inodescan_with_xattrs(gpfs_fssnap_handle_t *fssnapHandle,
                                              const gpfs_fssnap_id_t *prev_fssnapId,
                                              int nxAttrs,
                                              const char *xattrsList[],
                                              gpfs_ino_t *maxIno);
```

Description

The `gpfs_open_inodescan_with_xattrs()` subroutine opens an inode file and extended attributes for an inode scan identified by the `fssnapHandle` parameter. The scan traverses all user files, directories and links in the file system or snapshot. The scan begins with the user file with the lowest inode number and returns the files in increasing order. The `gpfs_seek_inode()` subroutine can be used to set the scan position to an arbitrary inode. System files, such as the block allocation maps, are omitted from the scan. The file system must be mounted to open an inode scan.

For an overview of using `gpfs_open_inodescan_with_xattrs()` in a backup application, see *Using APIs to develop backup applications* in *IBM Spectrum Scale: Administration Guide*.

To generate a full backup, invoke `gpfs_open_inodescan_with_xattrs()` with NULL for the `prev_fssnapId` parameter. Repeated invocations of `gpfs_next_inode()` then return inode information about all existing user files, directories, and links in inode number order.

To generate an incremental backup, invoke `gpfs_open_inodescan_with_xattrs()` with the `fssnapId` that was obtained from a `fssnapHandle` at the time the previous backup was created. The snapshot that was used for the previous backup does not need to exist at the time the incremental backup is generated. That is, the backup application needs to remember only the `fssnapId` of the previous backup; the snapshot itself can be deleted as soon as the backup is completed.

For the incremental backup, any operation that changes the file's `mtime` or `ctime` causes the file to be included. Files with no changes to the file's data or file attributes, other than a change to `atime`, are omitted from the scan.

A full inode scan (`prev_fssnapId` set to NULL) returns all inodes of existing files. An incremental inode scan (`prev_fssnapId` not NULL) returns inodes for files that have changed since the previous snapshot. The inodes of deleted files have a link count of zero.

If the snapshot indicated by `prev_fssnapId` is available, the caller may benefit from the extended read subroutine, `gpfs_ireadx()`, which returns only the changed blocks within the files. Without the previous snapshot, all blocks within the changed files are returned.

Once a full or incremental backup completes, the `new_fssnapId` must be saved in order to reuse it on a subsequent incremental backup. This `fssnapId` must be provided to the `gpfs_open_inodescan_with_xattrs()` subroutine, as the `prev_fssnapId` input parameter.

To read only the inodes that have been copied to a snapshot, use `gpfs_open_inodescan()` with `fssnapHandle` of the snapshot and pass `fssnapid` of the `fssnapHandle` as the `prev_fssnapId`.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fssnapHandle
The file system snapshot handle.

prev_fssnapId
A pointer to file system snapshot ID or NULL. If `prev_fssnapId` is provided, the inode scan returns only the files that have changed since the previous backup. If the pointer is NULL, the inode scan returns all user files.

If it is the same as the fssnapid of the fssnapHandle parameter, the scan only returns the inodes copied into the corresponding snapshot.

nxAttrs
The count of extended attributes to be returned. If `nxAttrs` is set to 0, call returns no extended attributes, like `gpfs_open_inodescan()`. If `nxAttrs` is set to `-1`, call returns all extended attributes.

xattrsList
A pointer to an array of pointers to names of extended attributes to be returned. `nxAttrsList` may be null if `nxAttrs` is set to 0 or `-1`.

maxIno
A pointer to inode number or NULL. If provided, `gpfs_open_inodescan_with_xattrs()` returns the maximum inode number in the file system or snapshot being scanned.

Exit status

If the `gpfs_open_inodescan_with_xattrs()` subroutine is successful, it returns a pointer to `gpfs_iscan_t`.

If the `gpfs_open_inodescan_with_xattrs()` subroutine is unsuccessful, it returns a NULL pointer and the global error variable `errno` is set to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EDOM
The file system snapshot ID passed for `prev_fssnapId` is from a different file system.

EINVAL
Incorrect parameters.

ENOMEM
Unable to allocate memory for the request.

ENOSYS
The `gpfs_open_inodescan_with_xattrs()` subroutine is not available.

EPERM
The caller does not have superuser privileges.

ERANGE
The `prev_fssnapId` parameter is the same as or more recent than `snapId` being scanned.

ESTALE
The cached file system information was not valid.

GPFS_E_INVAL_FSSNAPHANDLE
The file system snapshot handle is not valid.

GPFS_E_INVAL_FSSNAPID
The file system snapshot ID passed for `prev_fssnapId` is not valid.

Note: `gpfs_open_inodescan_with_xattrs()` calls the standard library subroutines `dup()` and `malloc()`; if one of these called subroutines returns an error, `gpfs_open_inodescan_with_xattrs()` also returns that error.

Examples

For an example using `gpfs_open_inodescan_with_xattrs()`, see `/usr/lpp/mmfs/samples/util/tsinode.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_open_inodescan_with_xattrs64() subroutine

Opens an inode file and extended attributes for an inode scan.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
gpfs_iscan_t *gpfs_open_inodescan_with_xattrs64(gpfs_fssnap_handle_t *fssnapHandle,
                                                const gpfs_fssnap_id_t *prev_fssnapId,
                                                int nxAttrs,
                                                const char *xattrList[],
                                                gpfs_ino64_t *maxIno);
```

Description

The `gpfs_open_inodescan_with_xattrs64()` subroutine opens an inode file and extended attributes for an inode scan identified by the `fssnapHandle` parameter. The scan traverses all user files, directories and links in the file system or snapshot. The scan begins with the user file with the lowest inode number and returns the files in increasing order. The `gpfs_seek_inode64()` subroutine may be used to set the scan position to an arbitrary inode. System files, such as the block allocation maps, are omitted from the scan. The file system must be mounted to open an inode scan.

For an overview of using `gpfs_open_inodescan_with_xattrs64()` in a backup application, see *Using APIs to develop backup applications* in *IBM Spectrum Scale: Administration Guide*.

To generate a full backup, invoke `gpfs_open_inodescan_with_xattrs64()` with NULL for the `prev_fssnapId` parameter. Repeated invocations of `gpfs_next_inode64()` then return inode information about all existing user files, directories, and links in inode number order.

To generate an incremental backup, invoke `gpfs_open_inodescan_with_xattrs64()` with the `fssnapId` that was obtained from a `fssnapHandle` at the time the previous backup was created. The snapshot that was used for the previous backup does not need to exist at the time the incremental backup is generated. That is, the backup application needs to remember only the `fssnapId` of the previous backup; the snapshot itself can be deleted as soon as the backup is completed.

For the incremental backup, any operation that changes the file's `mtime` or `ctime` causes the file to be included. Files with no changes to the file's data or file attributes, other than a change to `atime`, are omitted from the scan.

A full inode scan (`prev_fssnapId` set to NULL) returns all inodes of existing files. An incremental inode scan (`prev_fssnapId` not NULL) returns inodes for files that have changed since the previous snapshot. The inodes of deleted files have a link count of zero.

If the snapshot indicated by `prev_fssnapId` is available, the caller may benefit from the extended read subroutine, `gpfs_ireadx()`, which returns only the changed blocks within the files. Without the previous snapshot, all blocks within the changed files are returned.

Once a full or incremental backup completes, the `new_fssnapId` must be saved in order to reuse it on a subsequent incremental backup. This `fssnapId` must be provided to the `gpfs_open_inodescan_with_xattrs64()` subroutine, as the `prev_fssnapId` input parameter.

To read only the inodes that have been copied to a snapshot, use `gpfs_open_inodescan()` with `fssnapHandle` of the snapshot and pass `fssnapid` of the `fssnapHandle` as the `prev_fssnapId`.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fssnapHandle
The file system snapshot handle.

prev_fssnapId
A pointer to file system snapshot ID or NULL. If `prev_fssnapId` is provided, the inode scan returns only the files that have changed since the previous backup. If the pointer is NULL, the inode scan returns all user files.

If it is same as the fssnapid of the fssnapHandle parameter, the scan only returns the inodes copied into the corresponding snapshot.

nxAttrs
The count of extended attributes to be returned. If `nxAttrs` is set to 0, call returns no extended attributes, like `gpfs_open_inodescan64()`. If `nxAttrs` is set to `-1`, call returns all extended attributes

xattrsList
A pointer to an array of pointers to names of extended attributes to be returned. `nxAttrsList` may be null if `nxAttrs` is set to 0 or `-1`.

maxIno
A pointer to inode number or NULL. If provided, `gpfs_open_inodescan_with_xattrs64()` returns the maximum inode number in the file system or snapshot being scanned.

Exit status

If the `gpfs_open_inodescan_with_xattrs64()` subroutine is successful, it returns a pointer to `gpfs_iscan_t`.

If the `gpfs_open_inodescan_with_xattrs64()` subroutine is unsuccessful, it returns a NULL pointer and the global error variable `errno` is set to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EDOM
The file system snapshot ID passed for `prev_fssnapId` is from a different file system.

EINVAL
Incorrect parameters.

ENOMEM
Unable to allocate memory for the request.

ENOSYS
The `gpfs_open_inodescan_with_xattrs64()` subroutine is not available.

EPERM
The caller does not have superuser privileges.

ERANGE
The `prev_fssnapId` parameter is the same as or more recent than `snapId` being scanned.

ESTALE
The cached file system information was not valid.

GPFS_E_INVAL_FSSNAPHANDLE
The file system snapshot handle is not valid.

GPFS_E_INVAL_FSSNAPID
The file system snapshot ID passed for prev_fssnapId is not valid.

Note: gpfs_open_inodescan_with_xattrs64() calls the standard library subroutines dup() and malloc(); if one of these called subroutines returns an error, gpfs_open_inodescan_with_xattrs64() also returns that error.

Examples

See the gpfs_open_inodescan_with_xattrs() example in /usr/lpp/mmfs/samples/util/tsinode.c.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX

/usr/lpp/mmfs/lib/libgpfs.so for Linux

gpfs_prealloc() subroutine

Preallocates storage for a regular file or preallocates directory entries for a directory.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_prealloc(gpfs_file_t fileDesc,
                  gpfs_off64_t startOffset,
                  gpfs_off64_t bytesToPrealloc);
```

Description

The `gpfs_prealloc()` subroutine preallocates disk storage for a file or a directory.

In the case of a regular file, preallocation can improve I/O performance by creating a block of available disk storage in the file immediately instead of increasing the file size incrementally as data is written. The preallocated disk storage begins at the specified offset in the file and extends for the specified number of bytes, rounded up to a GPFS block boundary. Existing data is not modified. Reading any of the preallocated blocks returns zeroes. To determine how much space was actually preallocated, call the `stat()` subroutine and compare the reported file size and number of blocks used with their values before the preallocation.

In the case of a directory, preallocation can improve metadata performance by setting the minimum compaction size of the directory. Preallocation is most effective in systems in which many files are added to and removed from a directory in a short time. The minimum compaction size is the number of directory slots, including both full and empty slots, that a directory is allowed to retain when it is compacted. In IBM Spectrum Scale 4.1 or later versions, by default, a directory is automatically compacted as far as possible.

The `gpfs_prealloc()` subroutine sets the minimum compaction size of a directory to the specified number of slots and adds directory slots if needed to reach the minimum size. For example, if a directory contains 5,000 files and you set the minimum compaction size to 50,000, then the file system adds 45,000 directory slots. The directory can grow beyond 50,000 entries, but the file system does not allow the directory to be compacted below 50,000 slots.

You must specify the minimum compaction size as a number of bytes. Determine the number of bytes to allocate with the following formula:

```
bytesToPrealloc = n * ceiling((namelen + 13) / 32) * 32
```

where:

n
Specifies the number of directory entries that you want.

ceiling()
Is a function that rounds a fractional number up to the next highest integer. For example, ceiling(1.03125) returns 2.

namelen
Specifies the expected average length of file names.

For example, if you want 20,000 entries with an average file name length of 48, then bytesToPrealloc = 20000 * 2 * 32 = 1,280,000.

To restore the default behavior of the file system for a directory, in which a directory is compacted as far as possible, call the subroutine with `bytesToPrealloc` set to 0.

The number of bytes that are allocated for a directory is stored in the `ia_dirminsize` field in the `gpfs_iattr64_t` structure that is returned by the `gpfs_fstat_x()` subroutine. To convert the number of bytes to the number of directory slots, apply the following rule:

```
numSlots = numBytesReturned / 32
```

To convert the number of bytes to the number of files in the directory, apply a version of the formula that is described above:

```
numFiles = numBytesReturned / (ceiling(namelen + 13) * 32)
```

If the file is not a directory or is a directory with no preallocation, the `ia_dirminsize` field is set to 0. This attribute is reported for information only and is meaningful only in the active file system. It is not backed up and restored and it is ignored in snapshots. It usually differs from the file size that is recorded in `ia_size` or returned by `stat()`.

To set the minimum compaction size of a directory from the command line, set the `compact` parameter in the `mmchattr` command.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

fileDesc
The file descriptor returned by `open()`. Note the following points:

- A file must be opened for writing.
- A directory can be opened for reading, but the caller must have write permission to the directory.

startOffset
For a file, the byte offset into the file at which to begin preallocation. For a directory, set this parameter to 0.

bytesToPrealloc
The number of bytes to be preallocated. For a file, this value is rounded up to a GPFS block boundary. For a directory, calculate the number of bytes with the formula that is described above.

Exit status

If the `gpfs_prealloc()` subroutine is successful, it returns a value of 0.

If the `gpfs_prealloc()` subroutine is unsuccessful, it returns a value of `-1` and sets the global error variable `errno` to indicate the nature of the error. If `errno` is set to one of the following, some storage may have been preallocated:

- `EDQUOT`
- `ENOSPC`

The pre-allocation or minimum compaction setting of a directory can be obtained using the “gpfs_fstat_x() subroutine” on page 876. The gpfs_iattr64 structure it returns, defined in gpfs.h, is extended to include ia_dirminsize giving the pre-allocation size of a directory. For non-directories or directories without a pre-allocation set the value is zero. The size is expressed in bytes, which is interpreted in the same way as for the gpfs_prealloc() function. This will be 32 times the value reported by mmlsattr and will generally differ from the file size reported in ia_size and by stat().

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EACCES
The file or directory is not opened for writing.

EBADF
The file descriptor is not valid.

EDQUOT
A disk quota has been exceeded

EINVAL
The file descriptor does not refer to a file or directory; a negative value was specified for `startOffset` or `bytesToPrealloc`.

ENOSPC
The file system has run out of disk space.

ENOSYS
The `gpfs_prealloc()` subroutine is not supported under the current file system format.

Examples

```
#include <stdio.h>
#include <fcntl.h>
#include <errno.h>
#include <gpfs.h>

int rc;
int fileHandle = -1;
char* fileNameP = "datafile";
offset_t startOffset = 0;
offset_t bytesToAllocate = 20*1024*1024;  /* 20 MB */

fileHandle = open(fileNameP, O_RDWR|O_CREAT, 0644);
if (fileHandle < 0)
 {
   perror(fileNameP);
   exit(1);
 }

rc = gpfs_prealloc(fileHandle, startOffset, bytesToAllocate);
if (rc < 0)
 {
   fprintf(stderr, "Error %d preallocation at %lld for %lld in %s\n",
             errno, startOffset, bytesToAllocate, fileNameP);
   exit(1);
 }
```

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_putacl() subroutine

Restores the access control information for a GPFS file.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_putacl(const char *pathname,
                int flags,
                void *acl);
```

Description

The `gpfs_putacl()` subroutine together with the `gpfs_getacl()` subroutine is intended for use by a backup program to save (`gpfs_getacl()`) and restore (`gpfs_putacl()`) the ACL information for the file.

Notes:

1. The use of `gpfs_fgetattrs()` and `gpfs_fputattrs()` is preferred.
2. You must have `write` access to the file.
3. Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:
   - `libgpfs.a` for AIX
   - `libgpfs.so` for Linux

Parameters

pathname
Path name of the file for which the ACLs is to be set.

flags
Consists of one of these values:

0
Indicates that the `acl` parameter is to be mapped with the `gpfs_opaque_acl_t` structure.

The `gpfs_opaque_acl_t` structure should be used by backup and restore programs.

GPFS_PUTACL_STRUCT
Indicates that the `acl` parameter is to be mapped with the `gpfs_acl_t` structure.

The `gpfs_acl_t` structure is provided for applications that need to change the ACL.

acl
Pointer to a buffer mapped by the structure `gpfs_opaque_acl_t` or `gpfs_acl_t`, depending on the value of `flags`.

This is where the ACL data is stored, and should be the result of a previous invocation of `gpfs_getacl()`.

Exit status

If the `gpfs_putacl()` subroutine is successful, it returns a value of 0.

If the `gpfs_putacl()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EINVAL
The path name does not refer to a GPFS file or a regular file.

ENOMEM
Unable to allocate memory for the request.

ENOSYS
The `gpfs_putacl()` subroutine is not supported under the current file system format.

ENOTDIR
File is not a directory.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_putacl_fd() subroutine

Restores the access control information for a GPFS file.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_putacl_fd(gpfs_file_t fileDesc,
                   int flags,
                   void *acl);
```

Description

The `gpfs_putacl_fd()` subroutine together with the `gpfs_getacl_fd()` subroutine is intended for use by a backup program to save (`gpfs_getacl_fd()`) and restore (`gpfs_putacl_fd()`) the ACL information for the file.

Notes:

1. The use of `gpfs_fgetattrs()` and `gpfs_fputattrs()` is preferred.
2. You must have `write` access to the file.
3. Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:
   - `libgpfs.a` for AIX
   - `libgpfs.so` for Linux

Parameters

fileDesc
A file descriptor that identifies the file for which the ACLs are to be put.

flags
Consists of one of these values:

0
Indicates that the `acl` parameter is to be mapped with the `gpfs_opaque_acl_t` structure.

The `gpfs_opaque_acl_t` structure is used by backup and restore programs.

GPFS_PUTACL_STRUCT
Indicates that the `acl` parameter is to be mapped with the `gpfs_acl_t` structure.

The `gpfs_acl_t` structure is provided for applications that need to change the ACL.

acl
Pointer to a buffer mapped by the structure `gpfs_opaque_acl_t` or `gpfs_acl_t`, depending on the value of `flags`.

ACL data is stored here and should be the result of a previous invocation of `gpfs_getacl()`.

Exit status

If the `gpfs_putacl_fd()` subroutine is successful, it returns a value of 0.

If the `gpfs_putacl_fd()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following list:

Error codes include but are not limited to the following:

EINVAL
The file descriptor does not refer to a GPFS file or a regular file.

EBADF
The file descriptor is not valid.

ENOMEM
Unable to allocate memory for the request.

ENOSYS
The `gpfs_putacl_fd()` subroutine is not supported under the current file system format.

ENOTDIR
File is not a directory.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_quotactl() subroutine

Manipulates disk quotas on file systems.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_quotactl(const char *pathname,
                  int cmd,
                  int id,
                  void *bufferP);
```

Description

The `gpfs_quotactl()` subroutine manipulates disk quotas. It enables, disables, and manipulates disk quotas for file systems on which quotas have been enabled.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

pathname
Specifies the path name of any file within the mounted file system to which the quota control command is to applied.

cmd
Specifies the quota control command to be applied and whether it is applied to a user, group, or fileset quota.

The cmd parameter can be constructed using `GPFS_QCMD(qcmd, Type)` contained in `gpfs.h`. The `qcmd` parameter specifies the quota control command. The `Type` parameter specifies one of the following quota types:

- user (`GPFS_USRQUOTA`)
- group (`GPFS_GRPQUOTA`)
- fileset (`GPFS_FILESETQUOTA`)

The valid values for the `qcmd` parameter specified in `gpfs.h` are:

Q_QUOTAON
Enables quotas.

Enables disk quotas for the file system specified by the `pathname` parameter and type specified in `Type`. The `id` and `bufferP` parameters are unused. Root user authority is required to enable quotas.

Q_QUOTAOFF
Disables quotas.

Disables disk quotas for the file system specified by the `pathname` parameter and type specified in `Type`. The `id` and `bufferP` parameters are unused. Root user authority is required to disable quotas.

Q_GETQUOTA
Gets quota limits and usage information.

Retrieves quota limits and current usage for a user, group, or fileset specified by the id parameter. The bufferP parameter points to a gpfs_quotaInfo_t structure to hold the returned information. The gpfs_quotaInfo_t structure is defined in gpfs.h.

Root authority is required if the id value is not the current id (user id for GPFS_USRQUOTA, group id for GPFS_GRPQUOTA) of the caller.

Q_SETQUOTA
Sets quota limits

Sets disk quota limits for a user, group, or fileset specified by the id parameter. The bufferP parameter points to a gpfs_quotaInfo_t structure containing the new quota limits. The gpfs_quotaInfo_t structure is defined in gpfs.h. Root user authority is required to set quota limits.

Q_SETUSE
Sets quota usage

Sets disk quota usage for a user, group, or fileset specified by the id parameter. The bufferP parameter points to a gpfs_quotaInfo_t structure containing the new quota usage. The gpfs_quotaInfo_t structure is defined in gpfs.h. Root user authority is required to set quota usage.

Q_SYNC
Synchronizes the disk copy of a file system quota

Updates the on disk copy of quota usage information for a file system. The id and bufferP parameters are unused. Root user authority is required to synchronize a file system quota.

id
Specifies the user, group, or fileset ID to which the quota control command applies. The id parameters is interpreted by the specified quota type.

bufferP
Points to the address of an optional, command-specific data structure that is copied in or out of the system.

Exit status

If the gpfs_quotactl() subroutine is successful, it returns a value of 0.

If the gpfs_quotactl() subroutine is unsuccessful, it returns a value of -1 and sets the global error variable errno to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EACCES
Search permission is denied for a component of a path prefix.

EFAULT
An invalid bufferP parameter is supplied. The associated structure could not be copied in or out of the kernel.

EINVAL
One of the following errors:

- The file system is not mounted.
- Invalid command or quota type.

- Invalid input limits: negative limits or soft limits are greater than hard limits.
- UID is not defined.

ENOENT
No such file or directory.

EPERM
The quota control command is privileged and the caller did not have root user authority.

GPFS_E_NO_QUOTA_INST
The file system does not support quotas. This is the actual `errno` generated by GPFS.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_quotaInfo_t structure

Contains buffer mapping for the `gpfs_quotactl()` subroutine.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Structure

```
typedef struct gpfs_quotaInfo
{
  gpfs_off64_t blockUsage;      /* current block count */
  gpfs_off64_t blockHardLimit;  /* absolute limit on disk blks alloc */
  gpfs_off64_t blockSoftLimit;  /* preferred limit on disk blks */
  gpfs_off64_t blockInDoubt;    /* distributed shares + "lost" usage for blks */
  int          inodeUsage;      /* current # allocated inodes */
  int          inodeHardLimit;  /* absolute limit on allocated inodes */
  int          inodeSoftLimit;  /* preferred inode limit */
  int          inodeInDoubt;    /* distributed shares + "lost" usage for inodes */
  gpfs_uid_t   quoId;           /* uid, gid or fileset id
  int          entryType;       /* entry type, not used */
  unsigned int blockGraceTime;  /* time limit for excessive disk use */
  unsigned int inodeGraceTime;  /* time limit for excessive inode use */
} gpfs_quotaInfo_t;
```

Description

The `gpfs_quotaInfo_t` structure contains detailed information for the `gpfs_quotactl()` subroutine.

Members

blockUsage
The current block count in 1 KB units.

blockHardLimit
The absolute limit on disk block allocation.

blockSoftLimit
The preferred limit on disk block allocation.

blockInDoubt
The distributed shares and block usage that have not been not accounted for.

inodeUsage
The current number of allocated inodes.

inodeHardLimit
The absolute limit on allocated inodes.

inodeSoftLimit
The preferred inode limit.

inodeInDoubt
The distributed inode share and inode usage that have not been accounted for.

quoId
The user ID, group ID, or fileset ID.

entryType
Not used

blockGraceTime
The time limit (in seconds since the Epoch) for excessive disk use.

inodeGraceTime
The time limit (in seconds since the Epoch) for excessive inode use.

Epoch is midnight on January 1, 1970 UTC (Coordinated Universal Time).

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_seek_inode() subroutine

Advances an inode scan to the specified inode number.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_seek_inode(gpfs_iscan_t *iscan,
                    gpfs_ino_t ino);
```

Description

The `gpfs_seek_inode()` subroutine advances an inode scan to the specified inode number.

The `gpfs_seek_inode()` subroutine is used to start an inode scan at some place other than the beginning of the inode file. This is useful to restart a partially completed backup or an interrupted dump transfer to a mirror. It could also be used to do an inode scan in parallel from multiple nodes, by partitioning the inode number space into separate ranges for each participating node. The maximum inode number is returned when the scan was opened and each invocation to obtain the next inode specifies a termination inode number to avoid returning the same inode more than once.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

iscan
Pointer to the inode scan handle.

ino
The next inode number to be scanned.

Exit status

If the `gpfs_seek_inode()` subroutine is successful, it returns a value of 0.

If the `gpfs_seek_inode()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

ENOSYS
The `gpfs_seek_inode()` subroutine is not available.

GPFS_E_INVAL_ISCAN
Incorrect parameters.

Examples

For an example using `gpfs_seek_inode()`, see `/usr/lpp/mmfs/samples/util/tsinode.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_seek_inode64() subroutine

Advances an inode scan to the specified inode number.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_seek_inode64(gpfs_iscan_t *iscan,
                      gpfs_ino64_t ino);
```

Description

The `gpfs_seek_inode64()` subroutine advances an inode scan to the specified inode number.

The `gpfs_seek_inode64()` subroutine is used to start an inode scan at some place other than the beginning of the inode file. This is useful to restart a partially completed backup or an interrupted dump transfer to a mirror. It could also be used to do an inode scan in parallel from multiple nodes, by partitioning the inode number space into separate ranges for each participating node. The maximum inode number is returned when the scan was opened and each invocation to obtain the next inode specifies a termination inode number to avoid returning the same inode more than once.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

iscan
A pointer to the inode scan handle.

ino
The next inode number to be scanned.

Exit status

If the `gpfs_seek_inode64()` subroutine is successful, it returns a value of 0.

If the `gpfs_seek_inode64()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

ENOSYS
The `gpfs_seek_inode64()` subroutine is not available.

GPFS_E_INVAL_ISCAN
Incorrect parameters.

Examples

See the `gpfs_seek_inode()` example in `/usr/lpp/mmfs/samples/util/tsinode.c`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_stat() subroutine

Returns exact file status for a GPFS file.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_stat(const char *pathname,
              gpfs_stat64_t *buffer);
```

Description

The `gpfs_stat()` subroutine is used to obtain exact information about the file named by the `pathname` parameter. This subroutine is provided as an alternative to the `stat()` subroutine, which may not provide exact `mtime` and `atime` values. For more information, see *Exceptions to Open Group technical standards* in *IBM Spectrum Scale: Administration Guide*.

`read`, `write`, or `execute` permission for the named file is not required, but all directories listed in the path leading to the file must be searchable. The file information is written to the area specified by the `buffer` parameter.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

pathname
The path identifying the file for which exact status information is requested.

buffer
A pointer to the `gpfs_stat64_t` structure in which the information is returned. The `gpfs_stat64_t` structure is described in the `sys/stat.h` file.

Exit status

If the `gpfs_stat()` subroutine is successful, it returns a value of 0.

If the `gpfs_stat()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EINVAL
The path name does not refer to a GPFS file or a regular file.

ENOENT
The file does not exist.

ENOSYS
The `gpfs_stat()` subroutine is not supported under the current file system format.

ESTALE
The cached file system information was not valid.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_stat_inode() subroutine

Seeks the specified inode and retrieves that inode and its extended attributes from the inode scan.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_stat_inode(gpfs_iscan_t *iscan,
                    gpfs_ino_t ino,
                    gpfs_ino_t termIno,
                    const gpfs_iattr_t **iattr);
```

Description

The `gpfs_stat_inode()` subroutine is used to seek the specified inode and to retrieve that inode and its extended attributes from the inode scan. This subroutine combines `gpfs_seek_inode()` and `get_next_inode()`, but will only return the specified inode.

The data returned by `gpfs_next_inode()` is overwritten by subsequent calls to `gpfs_next_inode()`, `gpfs_seek_inode()`, or `gpfs_stat_inode()`.

The `termIno` parameter provides a way to partition an inode scan so it can be run on more than one node. It is only used by this call to control prefetching.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

iscan
A pointer to an inode scan descriptor.

ino
The inode number to be returned.

termIno
Prefetches inodes up to this inode. The caller might specify `maxIno` from `gpfs_open_inodescan()` or 0 to allow prefetching over the entire inode file.

iattr
A pointer to the returned pointer to the file's `iattr`.

Exit status

If the `gpfs_stat_inode()` subroutine is successful, it returns a value of 0 and the `iattr` parameter is set to point to `gpfs_iattr_t`. If the `gpfs_stat_inode()` subroutine is successful, but there are no more inodes before the `termIno` parameter, or if the requested inode does not exist, it returns a value of 0 and the `iattr` parameter is set to NULL.

If the `gpfs_stat_inode()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EPERM
The caller must have superuser privilege.

ENOSYS
The `gpfs_stat_inode()` subroutine is not supported under the current file system format.

ESTALE
The cached file system information was not valid.

ENOMEM
The buffer is too small.

GPFS_E_INVAL_ISCAN
Incorrect parameters.

GPFS_E_HOLE_IN_IFILE
The inode scan is reading only the inodes that have been copied to a snapshot and this inode has not yet been copied.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_stat_inode64() subroutine

Seeks the specified inode and retrieves that inode and its extended attributes from the inode scan.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_stat_inode64(gpfs_iscan_t *iscan,
                      gpfs_ino64_t ino,
                      gpfs_ino64_t termIno,
                      const gpfs_iattr64_t **iattr);
```

Description

The `gpfs_stat_inode64()` subroutine is used to seek the specified inode and to retrieve that inode and its extended attributes from the inode scan. This subroutine combines `gpfs_seek_inode64()` and `get_next_inode64()`, but will only return the specified inode.

The data returned by `gpfs_next_inode64()` is overwritten by subsequent calls to `gpfs_next_inode64()`, `gpfs_seek_inode64()`, or `gpfs_stat_inode64()`.

The `termIno` parameter provides a way to partition an inode scan so it can be run on more than one node. It is only used by this call to control prefetching.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

iscan
A pointer to an inode scan descriptor.

ino
The inode number to be returned.

termIno
Prefetches inodes up to this inode. The caller might specify `maxIno` from `gpfs_open_inodescan()` or 0 to allow prefetching over the entire inode file.

iattr
A pointer to the returned pointer to the file's `iattr`.

Exit status

If the `gpfs_stat_inode64()` subroutine is successful, it returns a value of 0 and the `iattr` parameter is set to point to `gpfs_iattr_t`.

If the `gpfs_stat_inode64()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EPERM
The caller must have superuser privilege.

ENOSYS
The `gpfs_stat_inode()` subroutine is not supported under the current file system format.

ESTALE
The cached file system information was not valid.

ENOMEM
The buffer is too small.

GPFS_E_INVAL_ISCAN
Incorrect parameters.

GPFS_E_HOLE_IN_IFILE
The inode scan is reading only the inodes that have been copied to a snapshot and this inode has not yet been copied.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_stat_inode_with_xattrs() subroutine

Seeks the specified inode and retrieves that inode and its extended attributes from the inode scan.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_stat_inode_with_xattrs(gpfs_iscan_t *iscan,
                                gpfs_ino_t ino,
                                gpfs_ino_t termIno,
                                const gpfs_iattr_t **iattr,
                                const char **xattrBuf,
                                unsigned int *xattrBufLen);
```

Description

The `gpfs_stat_inode_with_xattrs()` subroutine is used to seek the specified inode and to retrieve that inode and its extended attributes from the inode scan. This subroutine combines `gpfs_seek_inode()` and `get_next_inode()`, but will only return the specified inode.

The data returned by `gpfs_next_inode()` is overwritten by subsequent calls to `gpfs_next_inode()`, `gpfs_seek_inode()`, or `gpfs_stat_inode_with_xattrs()`.

The `termIno` parameter provides a way to partition an inode scan such that it can be run on more than one node. It is only used by this call to control prefetching.

The returned values for `xattrBuf` and `xattrBufLen` must be provided to `gpfs_next_xattr()` to obtain the extended attribute names and values. The buffer used for the extended attributes is overwritten by subsequent calls to `gpfs_next_inode()`, `gpfs_seek_inode()`, or `gpfs_stat_inode_with_xattrs()`.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

iscan
A pointer to an inode scan descriptor.

ino
The inode number to be returned.

termIno
Prefetches inodes up to this inode. The caller might specify `maxIno` from `gpfs_open_inodescan()` or 0 to allow prefetching over the entire inode file.

iattr
A pointer to the returned pointer to the file's `iattr`.

xattrBuf
A pointer to the returned pointer to the `xattr` buffer.

xattrBufLen
The returned length of the `xattr` buffer.

Exit status

If the `gpfs_stat_inode_with_xattrs()` subroutine is successful, it returns a value of 0 and the `iattr` parameter is set to point to `gpfs_iattr_t`. If the `gpfs_stat_inode_with_xattrs()` subroutine is successful, but there are no more inodes before the `termIno` parameter, or if the requested inode does not exist, it returns a value of 0 and the `iattr` parameter is set to NULL.

If the `gpfs_stat_inode_with_xattrs()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EPERM
The caller must have superuser privilege.

ENOSYS
The `gpfs_stat_inode_with_xattrs()` subroutine is not supported under the current file system format.

ESTALE
The cached file system information was not valid.

ENOMEM
The buffer is too small.

GPFS_E_INVAL_ISCAN
Incorrect parameters.

GPFS_E_HOLE_IN_IFILE
The inode scan is reading only the inodes that have been copied to a snapshot and this inode has not yet been copied.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_stat_inode_with_xattrs64() subroutine

Seeks the specified inode and retrieves that inode and its extended attributes from the inode scan.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_stat_inode_with_xattrs64(gpfs_iscan_t *iscan,
                                  gpfs_ino64_t ino,
                                  gpfs_ino64_t termIno,
                                  const gpfs_iattr64_t **iattr,
                                  const char **xattrBuf,
                                  unsigned int *xattrBufLen);
```

Description

The `gpfs_stat_inode_with_xattrs64()` subroutine is used to seek the specified inode and to retrieve that inode and its extended attributes from the inode scan. This subroutine combines `gpfs_seek_inode64()` and `get_next_inode64()`, but will only return the specified inode.

The data returned by `get_next_inode64()` is overwritten by subsequent calls to `gpfs_next_inode64()`, `gpfs_seek_inode64()`, or `gpfs_stat_inode_with_xattrs64()`.

The `termIno` parameter provides a way to partition an inode scan so it can be run on more than one node. It is only used by this call to control prefetching.

The returned values for `xattrBuf` and `xattrBufLen` must be provided to `gpfs_next_xattr()` to obtain the extended attribute names and values. The buffer used for the extended attributes is overwritten by subsequent calls to `gpfs_next_inode64()`, `gpfs_seek_inode64()`, or `gpfs_stat_inode_with_xattrs64()`.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.a` for AIX
- `libgpfs.so` for Linux

Parameters

iscan
A pointer to an inode scan descriptor.

ino
The inode number to be returned.

termIno
Prefetches inodes up to this inode. The caller might specify `maxIno` from `gpfs_open_inodescan64()` or 0 to allow prefetching over the entire inode file.

iattr
A pointer to the returned pointer to the file's `iattr`.

xattrBuf
A pointer to the returned pointer to the `xattr` buffer.

xattrBufLen
The returned length of the `xattr` buffer.

Exit status

If the `gpfs_stat_inode_with_xattrs64()` subroutine is successful, it returns a value of 0 and the `iattr` parameter is set to point to `gpfs_iattr_t`. If the `gpfs_stat_inode_with_xattrs64()` subroutine is successful, but there are no more inodes before the `termIno` parameter, or if the requested inode does not exist, it returns a value of 0 and the `iattr` parameter is set to NULL.

If the `gpfs_stat_inode_with_xattrs64()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following:

EPERM
The caller must have superuser privilege.

ENOSYS
The `gpfs_stat_inode_with_xattrs64()` subroutine is not supported under the current file system format.

ESTALE
The cached file system information was not valid.

ENOMEM
The buffer is too small.

GPFS_E_INVAL_ISCAN
Incorrect parameters.

GPFS_E_HOLE_IN_IFILE
The inode scan is reading only the inodes that have been copied to a snapshot and this inode has not yet been copied.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_stat_x() subroutine

Returns extended status information for a GPFS file with specified accuracy.

Library

GPFS library. Runs on Linux, AIX, and Windows.

Synopsis

```
#include <gpfs.h>
int gpfs_stat_x(const char *pathname,
            unsigned int *st_litemask,
            gpfs_iattr64_t *iattr,
            size_t iattrBufLen);
```

Description

The `gpfs_stat_x()` subroutine is similar to the `gpfs_stat()` subroutine but returns more information in a `gpfs_iattr64` structure that is defined in `gpfs.h`. This subroutine is supported only on the Linux operating system.

Your program must verify that the version of the `gpfs_iattr64` structure that is returned in the field `ia_version` is the same as the version that you are using. Versions are defined in `gpfs.h` with the pattern `GPFS_IA64_VERSION*`.

File permissions `read`, `write`, and `execute` are not required for the specified file, but all the directories in the specified path must be searchable.

Note: Compile any program that uses this subroutine with the `-lgpfs` flag from the following library:

- `libgpfs.so` for Linux

Parameters

***pathname**
A pointer to the path of the file for which information is requested.

***st_litemask**
A pointer to a bitmask specification of the items that you want to be returned exactly. Bitmasks are defined in `gpfs.h` with the pattern `GPFS_SLITE_*BIT`. The subroutine returns exact values for the items that you specify in the bitmask. The subroutine also sets bits in the bitmask to indicate any other items that are exact.

***iattr**
A pointer to a `gpfs_iattr64_t` structure in which the information is returned. The structure is described in the `gpfs.h` file.

iattrBufLen
The length of your `gpfs_iattr64_t` structure, as given by `sizeof(myStructure)`. The subroutine does not write data past this limit. The field `ia_reclen` in the returned structure is the length of the `gpfs_iattr64_t` structure that the system is using.

Exit status

If the `gpfs_stat_x()` subroutine is successful, it returns a value of 0.

If the `gpfs_stat_x()` subroutine is unsuccessful, it returns a value of -1 and sets the global error variable `errno` to indicate the nature of the error.

Exceptions

None.

Error status

Error codes include but are not limited to the following errors:

EINVAL
The path name does not refer to a GPFS file or a regular file.

ENOENT
The file does not exist.

ENOSYS
The `gpfs_stat_x()` subroutine is not supported under the current file system format.

ESTALE
The cached file system information was invalid.

Location

`/usr/lpp/mmfs/lib`

gpfsFcntlHeader_t structure

Contains declaration information for the `gpfs_fcntl()` subroutine.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Structure

```
typedef struct
{
  int     totalLength;
  int     fcntlVersion;
  int     errorOffset;
  int     fcntlReserved;
} gpfsFcntlHeader_t;
```

Description

The `gpfsFcntlHeader_t` structure contains size, version, and error information for the `gpfs_fcntl()` subroutine.

Members

totalLength
This field must be set to the total length, in bytes, of the data structure being passed in this subroutine. This includes the length of the header and all hints and directives that follow the header.

The total size of the data structure *cannot* exceed the value of `GPFS_MAX_FCNTL_LENGTH`, as defined in the header file `gpfs_fcntl.h`. The current value of `GPFS_MAX_FCNTL_LENGTH` is 64 KB.

fcntlVersion
This field must be set to the current version number of the `gpfs_fcntl()` subroutine, as defined by `GPFS_FCNTL_CURRENT_VERSION` in the header file `gpfs_fcntl.h`. The current version number is one.

errorOffset
If an error occurs processing a system call, GPFS sets this field to the offset within the parameter area where the error was detected.

For example,

1. An incorrect version number in the header would cause `errorOffset` to be set to zero.
2. An error in the first hint following the header would set `errorOffset` to `sizeof(header)`.

If no errors are found, GPFS does not alter this field.

fcntlReserved
This field is currently unused.

For compatibility with future versions of GPFS, set this field to zero.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfsGetDataBlkDiskIdx_t structure

Obtains the FPO data block location of a file.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Structure

```
typedef struct {
  int structLen;
  int structType;
  FilemapIn filemapIn;
  FilemapOut filemapOut;
} gpfsGetDataBlkDiskIdx_t;
```

Description

The `gpfsGetDataBlkDiskIdx_t` structure is used to obtain a file's blocks on-disk or in-memory location information.

Members

structLen
Length of the `gpfsGetDataBlkDiskIdx_t` structure.

structType
The structure identifiers are `GPFS_FCNTL_GET_DATABLKDISKIDX` and `GPFS_FCNTL_GET_DATABLKLOC`. `GPFS_FCNTL_GET_DATABLKDISKIDX` can be used to retrieve the disk ID on which the data blocks are located. `GPFS_FCNTL_GET_DATABLKLOC` can be used to retrieve the disk ID on which the data blocks are located and the node ID on which data block buffers (in memory) are located.

filemapIn
Input parameters:

```
struct FilemapIn {
    long long startOffset;
    long long skipfactor;
    long long length;
    int mreplicas;
    int reserved;
  } FilemapIn;
```

startOffset
Start offset in bytes.

skipfactor
Number of bytes to skip before the next offset read. Could be 0 in which case the meta block size will be used as the `skipfactor`.

length
Number of bytes to read; `startOffset` + `length` / `skipfactor` = `numBlksReturned`

mreplicas
Number of replicas that the user wants.

0 - all; 1 - primary; 2 - primary and 1st replica; 3 - primary, 1st and 2nd replica

reserved
Reserved field that should not be used.

filemapOut
Output data:

```
struct FilemapOut {
    int numReplicasReturned;
    int numBlksReturned;
    int blockSize;
    int blockSizeHigh;
    char buffer [GPFS_MAX_FCNTL_LENGTH-1024];
  } FilemapOut;
```

numReplicasReturned
Number of replicas returned.

numBlksReturned
Number of data blocks returned.

blockSize
Low 32bits of meta block size. Only valid if `skipfactor` in the input is 0. Otherwise, `blockSize` will be 0.

blockSizeHigh
High 32bits of meta block size. Only valid if `structType` is `GPFS_FCNTL_GET_DATABLKLOC` and `skipfactor` in the input is 0. Otherwise, `blockSizeHigh` will be 0.

buffer [GPFS_MAX_FCNTL_LENGTH-1024]
Buffer in which the disk ID and node ID are stored. If the buffer cannot store all of the requested information, you will have to calculate a new `startOffset` and `length` in the input data according to the output data and then repeat the call.

If `structType` is `GPFS_FCNTL_GET_DATABLKDISKIDX`, the format is:

```
If number of replicas returned is 1,
      offset(64bits) disid1(32bits)
      offset(64bits) diskid1(32bits)
      ...
      If number of replicas returned is 2,
      offset(64bits) diskid1(32bits) diskid2(32bits)
      offset(64bits) diskid1(32bits) diskid2(32bits)
      ...
      If number of replicas returned is 3,
      offset(64bits) diskid1(32bits) diskid2(32bits) diskid3(32bits)
      offset(64bits) diskid1(32bits) diskid2(32bits) diskid3(32bits)
      ...
```

If `structType` is `GPFS_FCNTL_GET_DATABLKLOC`, the format is:

```
If number of replicas returned is 1,
      disid1(32bits) nodeid1(32bits)
      disid1(32bits) nodeid1(32bits)
      ...
      If number of replicas returned is 2,
      disid1(32bits) diskid2(32bits) nodeid1(32bits) nodeid2(32bits)
      disid1(32bits) diskid2(32bits) nodeid1(32bits) nodeid2(32bits)
      ...
      If number of replicas returned is 3,
      disid1(32bits) diskid2(32bits) diskid3(32bits) nodeid1(32bits) nodeid2(32bits)
       nodeid3(32bits)
      disid1(32bits) diskid2(32bits) diskid3(32bits) nodeid1(32bits) nodeid2(32bits)
       nodeid3(32bits)
      ...
```

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfsGetFilesetName_t structure

Obtains the fileset name of a file.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Structure

```
typedef struct {
    int structLen;
    int structType;
    char buffer[GPFS_FCNTL_MAX_NAME_BUFFER];
} gpfsGetFilesetName_t;
```

Description

The `gpfsGetFilesetName_t` structure is used to obtain a file's fileset name.

Members

structLen
Length of the `gpfsGetFilesetName_t` structure.

structType
Structure identifier `GPFS_FCNTL_GET_FILESETNAME`.

buffer
The size of the buffer may vary, but must be a multiple of eight. Upon successful completion of the call, the buffer contains a null-terminated character string for the name of the requested object.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfsGetReplication_t structure

Obtains the replication factors of a file.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Structure

```
typedef struct {
    int structLen;
    int structType;
    int metadataReplicas;
    int maxMetadataReplicas;
    int dataReplicas;
    int maxDataReplicas;
    int status;
    int reserved;
} gpfsGetReplication_t;
```

Description

The `gpfsGetReplication_t` structure is used to obtain a file's replication factors.

Members

structLen
Length of the `gpfsGetReplication_t` structure.

structType
Structure identifier `GPFS_FCNTL_GET_REPLICATION`.

metadataReplicas
Returns the current number of copies of indirect blocks for the file.

maxMetadataReplicas
Returns the maximum number of copies of indirect blocks for a file.

dataReplicas
Returns the current number of copies of the data blocks for a file.

maxDataReplicas
Returns the maximum number of copies of data blocks for a file.

status
Returns the status of the file.

reserved
Unused, but should be set to 0.

Error status

These values are returned in the `status` field:

GPFS_FCNTL_STATUS_EXPOSED
This file may have some data where the only replicas are on suspended disks; implies some data may be lost if suspended disks are removed.

GPFS_FCNTL_STATUS_ILLREPLICATE
This file may not be properly replicated; that is, some data may have fewer or more than the desired number of replicas, or some replicas may be on suspended disks.

GPFS_FCNTL_STATUS_UNBALANCED
This file may not be properly balanced.

GPFS_FCNTL_STATUS_DATAUPDATEMISS
This file has stale data blocks on at least one of the disks that are marked as unavailable or recovering.

GPFS_FCNTL_STATUS_METAUPDATEMISS
This file has stale indirect blocks on at least one unavailable or recovering disk.

GPFS_FCNTL_STATUS_ILLPLACED
This file may not be properly placed; that is, some data may be stored in an incorrect storage pool.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfsGetSetXAttr_t structure

Obtains or sets extended attribute values.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Structure

```
typedef struct {
  int structLen;
  int structType;
  int nameLen;
  int bufferLen;
  unsigned int flags;
  int errReasonCode;
  char buffer[0];
} gpfsGetSetXAttr_t;
```

Description

The `gpfsGetSetXAttr_t` structure is used to obtain extended attributes.

Members

structLen
Length of the `gpfsGetSetXAttr_t` structure.

structType
Structure identifier `GPFS_FCNTL_GET_XATTR` or `GPFS_FCNTL_SET_XATTR`.

nameLen
Length of the attribute name. May include a trailing '\0' character.

bufferLen
For `GPFS_FCNTL_GET_XATTR`: Input, length of the buffer; output, length of the attribute value.

For `GPFS_FCNTL_SET_XATTR`: Input, length of the attribute value. Specify `-1` to delete an attribute.

errReasonCode
Reason code.

flags
The following flags are recognized:

- `GPFS_FCNTL_XATTRFLAG_NONE`
- `GPFS_FCNTL_XATTRFLAG_SYNC`
- `GPFS_FCNTL_XATTRFLAG_CREATE`
- `GPFS_FCNTL_XATTRFLAG_REPLACE`
- `GPFS_FCNTL_XATTRFLAG_DELETE`
- `GPFS_FCNTL_XATTRFLAG_NO_CTIME`
- `GPFS_FCNTL_XATTRFLAG_RESERVED`

buffer
Buffer for the attribute name and value.

For `GPFS_FCNTL_GET_XATTR`:

Input: The name begins at offset 0 and must be null terminated.
Output: The name is returned unchanged; the value begins at `nameLen` rounded up to a multiple of 8.

For GPFS_FCNTL_SET_XATTR:

Input: The name begins at offset 0 and must be null terminated. The value begins at nameLen rounded up to a multiple of 8.

The actual length of the buffer should be nameLen rounded up to a multiple of 8, plus the length of the attribute value rounded up to a multiple of 8.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX

/usr/lpp/mmfs/lib/libgpfs.so for Linux

gpfsGetSnapshotName_t structure

Obtains the snapshot name of a file.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Structure

```
typedef struct {
    int structLen;
    int structType;
    char buffer[GPFS_FCNTL_MAX_NAME_BUFFER];
} gpfsGetSnapshotName_t;
```

Description

The `gpfsGetSnapshotName_t` structure is used to obtain a file's snapshot name. If the file is not part of a snapshot, a zero length snapshot name will be returned.

Members

structLen
Length of the `gpfsGetSnapshotName_t` structure.

structType
Structure identifier `GPFS_FCNTL_GET_SNAPSHOTNAME`.

buffer
The size of the buffer may vary, but must be a multiple of eight. Upon successful completion of the call, the buffer contains a null-terminated character string for the name of the requested object.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfsGetStoragePool_t structure

Obtains the storage pool name of a file.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Structure

```
typedef struct {
    int structLen;
    int structType;
    char buffer[GPFS_FCNTL_MAX_NAME_BUFFER];
} gpfsGetStoragePool_t;
```

Description

The `gpfsGetStoragePool_t` structure is used to obtain a file's storage pool name.

Members

structLen
Length of the `gpfsGetStoragePool_t` structure.

structType
Structure identifier `GPFS_FCNTL_GET_STORAGEPOOL`.

buffer
The size of the buffer may vary, but must be a multiple of eight. Upon successful completion of the call, the buffer contains a null-terminated character string for the name of the requested object.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfsListXAttr_t structure

Lists extended attributes.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Structure

```
typedef struct {
  int structLen;
  int structType;
  int bufferLen;
  int errReasonCode;
  char buffer[0];
} gpfsListXAttr_t;
```

Description

The `gpfsListXAttr_t` structure is used to list extended attributes.

Members

structLen
Length of the `gpfsListXAttr_t` structure.

structType
Structure identifier `GPFS_FCNTL_LIST_XATTR` .

bufferLen
Input: Length of the buffer. Output: Length of the returned list of names.

The actual length of the buffer required depends on the number of attributes set and the length of each attribute name. If the buffer provided is too small for all of the returned names, the `errReasonCode` will be set to `GPFS_FCNTL_ERR_BUFFER_TOO_SMALL`, and `bufferLen` will be set to the minimum size buffer required to list all attributes. An initial buffer length of 0 may be used to query the attributes and determine the correct buffer size for this file.

errReasonCode
Reason code.

buffer
Buffer for the returned list of names. Each attribute name is prefixed with a one-byte name length. The attribute name may contain embedded null bytes. The attribute name may be null-terminated in which case the name length includes this null byte. The next attribute name follows immediately in the buffer (and is prefixed with its own length). Following the last name, a `'\0'` is appended to terminate the list. The returned `bufferLen` includes the final `'\0'`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfsRestripeData_t structure

Restripes the data blocks of a file.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Structure

```
typedef struct {
  int structLen;
  int structType;
  int options;
  int errReason;
  int errValue1;
  int errValue2;
  int reserved1;
  int reserved2;
} gpfsRestripeData_t;
```

Description

The `gpfsRestripeData_t` structure is used to restripe a file's data blocks to updates its replication and migrate its data. The data movement is always done immediately.

Members

structLen
Length of the `gpfsRestripeData_t` structure.

structType
Structure identifier `GPFS_FCNTL_RESTRIPE_DATA`.

options
Options for restripe command. See the `mmrestripefile` command for complete definitions.

GPFS_FCNTL_RESTRIPE_M
Migrate critical data off of suspended disks.

GPFS_FCNTL_RESTRIPE_R
Replicate data against subsequent failure.

GPFS_FCNTL_RESTRIPE_P
Place file data in assigned storage pool.

GPFS_FCNTL_RESTRIPE_B
Rebalance file data.

errReason
Reason code describing the failure. Possible codes are defined in “Error status” on page 998.

errValue1
Returned value depending upon `errReason`.

errValue2
Returned value depending upon `errReason`.

reserved1
Unused, but should be set to 0.

reserved2
Unused, but should be set to 0.

Error status

These values are returned in the `errReason` field:

GPFS_FCNTL_ERR_NO_REPLICA_GROUP
Not enough replicas could be created because the desired degree of replication is larger than the number of failure groups.

GPFS_FCNTL_ERR_NO_REPLICA_SPACE
Not enough replicas could be created because there was not enough space left in one of the failure groups.

GPFS_FCNTL_ERR_NO_BALANCE_SPACE
There was not enough space left on one of the disks to properly balance the file according to the current stripe method.

GPFS_FCNTL_ERR_NO_BALANCE_AVAILABLE
The file could not be properly balanced because one or more disks are unavailable.

GPFS_FCNTL_ERR_ADDR_BROKEN
All replicas were on disks that have since been deleted from the stripe group.

GPFS_FCNTL_ERR_NO_IMMUTABLE_DIR
No immutable attribute can be set on directories.

GPFS_FCNTL_ERR_NO_IMMUTABLE_SYSFILE
No immutable attribute can be set on system files.

GPFS_FCNTL_ERR_IMMUTABLE_FLAG
Immutable and indefinite retention flag is wrong.

GPFS_FCNTL_ERR_IMMUTABLE_PERM
Immutable and indefinite retention flag is wrong.

GPFS_FCNTL_ERR_APPENDONLY_CONFLICT
The `appendOnly` flag should be set separately.

GPFS_FCNTL_ERR_NOIMMUTABLE_ONSNAP
Cannot set immutable or `appendOnly` on snapshots.

GPFS_FCNTL_ERR_FILE_HAS_XATTRS
An attempt to change `maxDataReplicas` or `maxMetadataReplicas` was made on a file that has extended attributes.

GPFS_FCNTL_ERR_NOT_GPFS_FILE
This file is not part of a GPFS file system.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfsRestripeRange_t structure

Restripes a specific range of data blocks of a file.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Structure

```
typedef struct {
  int structLen;
  int structType;
  int options;
  int errReason;
  int errValue1;
  int errValue2;
  gpfsByteRange_t range;
  int reserved1;
  int reserved2;
} gpfsRestripeRange_t;
```

Description

The `gpfsRestripeRange_t` structure is used to restripe a specific range of data blocks of a file to update its replication and migrate its data. The data movement is always done immediately.

Members

structLen
Length of the `gpfsRestripeRange_t` structure.

structType
Structure identifier `GPFS_FCNTL_RESTRIPE_RANGE`.

options
Options for the restripe command. See the **mmrestripefile** command for complete definitions.

GPFS_FCNTL_RESTRIPE_M
Migrates critical data off suspended disks.

GPFS_FCNTL_RESTRIPE_R
Replicates data against subsequent failure.

GPFS_FCNTL_RESTRIPE_P
Places file data in assigned storage pool.

GPFS_FCNTL_RESTRIPE_B
Rebalances file data.

GPFS_FCNTL_RESTRIPE_L
Relocates file data.

GPFS_FCNTL_RESTRIPE_C
Compresses file data.

GPFS_FCNTL_RESTRIPE_RANGE_R
Indicates to restripe only a range. If it is not set, the whole file is restriped.

errReason
Provides a reason code that describes the failure. Possible codes are defined in the Error status.

errValue1
Returned value depending upon `errReason`.

errValue2
Returned value depending upon `errReason`.

range
Used with GPFS_FCNTL_RESTRIPE_RANGE_R to specify a range. Otherwise, it should be set to 0.

reserved1
Unused, but it should be set to 0.

reserved2
Unused, but it should be set to 0.

Error status
These values are returned in the errReason field:

GPFS_FCNTL_ERR_NO_REPLICA_GROUP
Not enough replicas could be created because the desired degree of replication is larger than the number of failure groups.

GPFS_FCNTL_ERR_NO_REPLICA_SPACE
Not enough replicas could be created because there was not enough space left in one of the failure groups.

GPFS_FCNTL_ERR_NO_BALANCE_SPACE
There was not enough space left on one of the disks to properly balance the file according to the current stripe method.

GPFS_FCNTL_ERR_NO_BALANCE_AVAILABLE
The file could not be properly balanced because one or more disks are unavailable.

GPFS_FCNTL_ERR_ADDR_BROKEN
All replicas were on disks that were deleted from the stripe group.

GPFS_FCNTL_ERR_NO_IMMUTABLE_DIR
No immutable attribute can be set on directories.

GPFS_FCNTL_ERR_NO_IMMUTABLE_SYSFILE
No immutable attribute can be set on system files.

GPFS_FCNTL_ERR_IMMUTABLE_FLAG
Immutable and indefinite retention flag is wrong.

GPFS_FCNTL_ERR_IMMUTABLE_PERM
Immutable and indefinite retention flag is wrong.

GPFS_FCNTL_ERR_APPENDONLY_CONFLICT
The appendOnly flag should be set separately.

GPFS_FCNTL_ERR_NOIMMUTABLE_ONSNAP
Cannot set immutable or appendOnly on snapshots.

GPFS_FCNTL_ERR_FILE_HAS_XATTRS
An attempt to change maxDataReplicas or maxMetadataReplicas was made on a file that has extended attributes.

GPFS_FCNTL_ERR_NOT_GPFS_FILE
This file is not part of a GPFS file system.

GPFS_FCNTL_ERR_COMPRESS_SNAPSHOT
Compression is not supported for snapshot files. Used by GPFS_FCNTL_RESTRIPE_C only.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX

/usr/lpp/mmfs/lib/libgpfs.so for Linux

gpfsRestripeRangeV2_t structure

Restripes a specific range of data blocks of a file.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Structure

```
typedef struct {
  int structLen;
  int structType;
  int options;
  int errReason;
  int errValue1;
  int errValue2;
  gpfsByteRange_t range;
  int reserved1;
  int reserved2;
  char *outBuf;
  int bufLen;
  int reserved3;
} gpfsRestripeRangeV2_t;
```

Description

The `gpfsRestripeRangeV2_t` structure is used to restripe a specific range of data blocks of a file to update its replication and migrate its data. The data movement is always done immediately. The difference between `gpfsRestripeRange_t` and `gpfsRestripeRangeV2_t` is that `gpfsRestripeRangeV2_t` has an `outBuf`, which could be used to pass information from the daemon back to the caller. Currently, only `GPFS_FCNTL_RESTRIPE_CM` uses the buffer.

Members

structLen
Length of the `gpfsRestripeRangeV2_t` structure.

structType
Structure identifier `GPFS_FCNTL_RESTRIPE_RANGE`.

options
Options for the restripe command. See the **mmrestripefile** command for complete definitions.

GPFS_FCNTL_RESTRIPE_M
Migrates critical data off suspended disks.

GPFS_FCNTL_RESTRIPE_R
Replicates data against subsequent failure.

GPFS_FCNTL_RESTRIPE_P
Places file data in assigned storage pool.

GPFS_FCNTL_RESTRIPE_B
Rebalances file data.

GPFS_FCNTL_RESTRIPE_L
Relocates file data.

GPFS_FCNTL_RESTRIPE_CM
Compares the content of replicas of file data.

GPFS_FCNTL_RESTRIPE_C
Compresses file data.

GPFS_FCNTL_RESTRIPE_RANGE_R
Indicates to restripe only a range. If it is not set, the whole file is restriped.

errReason
Provides a reason code that describes the failure. Possible codes are defined in the Error status.

errValue1
Returned value depending upon errReason.

errValue2
Returned value depending upon errReason.

range
Used with GPFS_FCNTL_RESTRIPE_RANGE_R to specify a range. Otherwise, it should be set to 0.

reserved1
Unused, but it should be set to 0.

reserved2
Unused, but it should be set to 0.

outBuf
Buffer to store returned information from the daemon. The caller is responsible for allocation and deallocation of the buffer. GPFS_FCNTL_RESTRIPE_CM only uses it to return bad replica information.

bufLen
Length of outBuf.

reserved3
Unused, but it should be set to 0.

Error status
These values are returned in the errReason field:

GPFS_FCNTL_ERR_NO_REPLICA_GROUP
Not enough replicas could be created because the desired degree of replication is larger than the number of failure groups.

GPFS_FCNTL_ERR_NO_REPLICA_SPACE
Not enough replicas could be created because there was not enough space left in one of the failure groups.

GPFS_FCNTL_ERR_NO_BALANCE_SPACE
There was not enough space left on one of the disks to properly balance the file according to the current stripe method.

GPFS_FCNTL_ERR_NO_BALANCE_AVAILABLE
The file could not be properly balanced because one or more disks are unavailable.

GPFS_FCNTL_ERR_ADDR_BROKEN
All replicas were on disks that were deleted from the stripe group.

GPFS_FCNTL_ERR_NO_IMMUTABLE_DIR
No immutable attribute can be set on directories.

GPFS_FCNTL_ERR_NO_IMMUTABLE_SYSFILE
No immutable attribute can be set on system files.

GPFS_FCNTL_ERR_IMMUTABLE_FLAG
Immutable and indefinite retention flag is wrong.

GPFS_FCNTL_ERR_IMMUTABLE_PERM
Immutable and indefinite retention flag is wrong.

GPFS_FCNTL_ERR_APPENDONLY_CONFLICT
The appendOnly flag should be set separately.

GPFS_FCNTL_ERR_NOIMMUTABLE_ONSNAP
Cannot set immutable or appendOnly on snapshots.

GPFS_FCNTL_ERR_FILE_HAS_XATTRS
An attempt to change maxDataReplicas or maxMetadataReplicas was made on a file that has extended attributes.

GPFS_FCNTL_ERR_NOT_GPFS_FILE
This file is not part of a GPFS file system.

GPFS_FCNTL_ERR_COMPRESS_SNAPSHOT
Compression is not supported for snapshot files. Used by GPFS_FCNTL_RESTRIPE_C only.

GPFS_FCNTL_ERR_COMPARE_DATA_IN_INODE
Replica comparison does not support data-in-inode file. Used by GPFS_FCNTL_RESTRIPE_CM only.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX

/usr/lpp/mmfs/lib/libgpfs.so for Linux

gpfsSetReplication_t structure

Sets the replication factors of a file.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Structure

```
typedef struct {
    int structLen;
    int structType;
    int metadataReplicas;
    int maxMetadataReplicas;
    int dataReplicas;
    int maxDataReplicas;
    int errReason;
    int errValue1;
    int errValue2;
    int reserved;
} gpfsSetReplication_t;
```

Description

The `gpfsGetReplication_t` structure is used to set a file's replication factors. However, the directive does not cause the file to be restriped immediately. Instead, the caller must append a `gpfsRestripeData_t` directive or invoke an explicit restripe using the `mmrestripefs` or `mmrestripefile` command.

Members

structLen
Length of the `gpfsSetReplication_t` structure.

structType
Structure identifier `GPFS_FCNTL_SET_REPLICATION`.

metadataReplicas
Specifies how many copies of the file system's metadata to create. Enter a value of 1 or 2, but not greater than the value of the `maxMetadataReplicas` attribute of the file. A value of 0 indicates not to change the current value.

maxMetadataReplicas
The maximum number of copies of indirect blocks for a file. Space is reserved in the inode for all possible copies of pointers to indirect blocks. Valid values are 1 and 2, but cannot be less than `DefaultMetadataReplicas`. The default is 1. A value of 0 indicates not to change the current value.

dataReplicas
Specifies how many copies of the file data to create. Enter a value of 1 or 2, but not greater than the value of the `maxDataReplicas` attribute of the file. A value of 0 indicates not to change the current value.

maxDataReplicas
The maximum number of copies of data blocks for a file. Space is reserved in the inode and indirect blocks for all possible copies of pointers to data blocks. Valid values are 1 and 2, but cannot be less than `DefaultDataReplicas`. The default is 1. A value of 0 indicates not to change the current value.

errReason
Reason code describing the failure. Possible codes are defined in "Error status" on page 1005.

errValue1
Returned value depending upon `errReason`.

errValue2
Returned value depending upon errReason.

reserved
Unused, but should be set to 0.

Error status

These values are returned in the errReason field:

GPFS_FCNTL_ERR_NONE
Command was successful or no reason information was returned.

GPFS_FCNTL_ERR_METADATA_REPLICAS_RANGE
Field metadataReplicas is out of range. Fields errValue1 and errValue2 contain the valid lower and upper range boundaries.

GPFS_FCNTL_ERR_MAXMETADATA_REPLICAS_RANGE
Field maxMetadataReplicas is out of range. Fields errValue1 and errValue2 contain the valid lower and upper range boundaries.

GPFS_FCNTL_ERR_DATA_REPLICAS_RANGE
Field dataReplicas is out of range. Fields errValue1 and errValue2 contain the valid lower and upper range boundaries.

GPFS_FCNTL_ERR_MAXDATA_REPLICAS_RANGE
Field maxDataReplicas is out of range. Fields errValue1 and errValue2 contain the valid lower and upper range boundaries.

GPFS_FCNTL_ERR_FILE_NOT_EMPTY
An attempt to change maxMetadataReplicas or maxDataReplicas or both was made on a file that is not empty.

GPFS_FCNTL_ERR_REPLICAS_EXCEED_FGMAX
Field metadataReplicas, or dataReplicas, or both exceed the number of failure groups. Field errValue1 contains the maximum number of metadata failure groups. Field errValue2 contains the maximum number of data failure groups.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX

/usr/lpp/mmfs/lib/libgpfs.so for Linux

gpfsSetStoragePool_t structure

Sets the assigned storage pool of a file.

Library

GPFS Library (`libgpfs.a` for AIX, `libgpfs.so` for Linux)

Structure

```
typedef struct {
    int structLen;
    int structType;
    int errReason;
    int errValue1;
    int errValue2;
    int reserved;
    char buffer[GPFS_FCNTL_MAX_NAME_BUFFER];
} gpfsSetStoragePool_t;
```

Description

The `gpfsSetStoragePool_t` structure is used to set a file's assigned storage pool. However, the directive does not cause the file data to be migrated immediately. Instead, the caller must append a `gpfsRestripeData_t` directive or invoke an explicit restripe with the `mmrestripefs` or `mmrestripefile` command. The caller must have su or root privileges to change a storage pool assignment.

Members

structLen
Length of the `gpfsSetStoragePool_t` structure.

structType
Structure identifier `GPFS_FCNTL_SET_STORAGEPOOL`.

errReason
Reason code describing the failure. Possible codes are defined in "Error status" on page 1006.

errValue1
Returned value depending upon `errReason`.

errValue2
Returned value depending upon `errReason`.

reserved
Unused, but should be set to 0.

buffer
The name of the storage pool for the file's data. Only user files may be reassigned to different storage pool. System files, including all directories, must reside in the system pool and may not be moved. The size of the buffer may vary, but must be a multiple of eight.

Error status

These values are returned in the `errReason` field:

GPFS_FCNTL_ERR_NONE
Command was successful or no reason information was returned.

GPFS_FCNTL_ERR_NOPERM
User does not have permission to perform the requested operation.

GPFS_FCNTL_ERR_INVALID_STORAGE_POOL
Invalid storage pool name was given.

GPFS_FCNTL_ERR_INVALID_STORAGE_POOL_TYPE
Invalid storage pool. File cannot be assigned to given pool.

GPFS_FCNTL_ERR_INVALID_STORAGE_POOL_ISDIR
Invalid storage pool. Directories cannot be assigned to given pool.

GPFS_FCNTL_ERR_INVALID_STORAGE_POOL_ISLNK
Invalid storage pool. System files cannot be assigned to given pool.

GPFS_FCNTL_ERR_INVALID_STORAGE_POOL_ISSYS
Invalid storage pool. System files cannot be assigned to given pool.

GPFS_FCNTL_ERR_STORAGE_POOL_NOTENABLED
File system has not been upgraded to support storage pools.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX

`/usr/lpp/mmfs/lib/libgpfs.so` for Linux

Chapter 4. GPFS user exits

Apart from the user exits defined by using the **mmaddcallback** command, GPFS provides four more user exits: **mmsdrbackup**, **nsddevices**, **syncfsconfig**, and **preunmount**.

Table 36 on page 1009 summarizes the GPFS-specific user exits.

Table 36. GPFS user exits

User exit	Purpose
"mmsdrbackup user exit" on page 1010	Performs a backup of the GPFS configuration data.
"nsddevices user exit" on page 1011	Identifies local physical devices that are used as GPFS Network Shared Disks (NSDs).
"syncfsconfig user exit" on page 1012	Keeps file system configuration data in replicated clusters synchronized.
"preunmount user exit" on page 1013	Helps execute a clean shutdown of IBM Spectrum Scale when callbacks are not effective.

mmsdrbackup user exit

Performs a backup of the GPFS configuration data.

Description

The `/var/mmfs/etc/mmsdrbackup` user exit, when properly installed on the primary GPFS configuration server, is called asynchronously every time there is a change to the GPFS master configuration file. You can use this user exit to create a backup of the GPFS configuration data.

Read the sample file `/usr/lpp/mmfs/samples/mmsdrbackup.sample` for a detailed description on how to code and install this user exit.

The type of backup that is created depends on the configuration of the cluster:

- If the Cluster Configuration Repository (CCR) is enabled, then a CCR backup is created. This type of backup applies to IBM Spectrum Scale 4.2.0 or later.
- Otherwise, a mmsdrfs backup is created.

For more information about the CCR, see "mmcrcluster command" on page 306.

Note: The `mmsdrbackup` user exit is supported from IBM Spectrum Scale 4.2.0 or later. It must not be used on clusters which have nodes running earlier versions of IBM Spectrum Scale.

Parameters

The generation number of the most recent version of the GPFS configuration data.

Exit status

The `mmsdrbackup` user exit returns a value of zero.

Location

`/var/mmfs/etc`

nsddevices user exit

Identifies local physical devices that are used as GPFS Network Shared Disks (NSDs).

Description

The `/var/mmfs/etc/nsddevices` user exit, when properly installed, is invoked synchronously by the GPFS daemon during its disk discovery processing. The purpose of this procedure is to discover and verify the physical devices on each node that correspond to the disks previously defined to GPFS with the `mmcrnsd` command. The `nsddevices` user exit can be used to either replace or to supplement the disk discovery procedure of the GPFS daemon.

Read the sample file `/usr/lpp/mmfs/samples/nsddevices.sample` for a detailed description on how to code and install this user exit.

Parameters

None.

Exit status

The `nsddevices` user exit should return either zero or one.

When the `nsddevices` user exit returns a value of zero, the GPFS disk discovery procedure is bypassed.

When the `nsddevices` user exit returns a value of one, the GPFS disk discovery procedure is performed and the results are concatenated with the results from the `nsddevices` user exit.

Location

`/var/mmfs/etc`

syncfsconfig user exit

Keeps file system configuration data in replicated clusters synchronized.

Description

The `/var/mmfs/etc/syncfsconfig` user exit, when properly installed, will be synchronously invoked after each command that may change the configuration of a file system. Examples of such commands are: `mmadddisk`, `mmdeldisk`, `mmchfs`, and so forth. The `syncfsconfig` user exit can be used to keep the file system configuration data in replicated GPFS clusters automatically synchronized.

Read the sample file `/usr/lpp/mmfs/samples/syncfsconfig.sample` for a detailed description on how to code and install this user exit.

Parameters

None.

Exit status

The `syncfsconfig` user exit should always return a value of zero.

Location

`/var/mmfs/etc`

preunmount user exit

Helps execute a clean shutdown of IBM Spectrum Scale when callbacks are not effective. For example, when bind mounts pointing inside a file system are in use.

Description

You can use the `/var/mmfs/etc/preunmount` user exit to execute a script before an IBM Spectrum Scale file system is unmounted.

Parameters

None.

Exit status

The **`preunmount`** user exit returns a value of zero.

Location

`/var/mmfs/etc`

Chapter 5. IBM Spectrum Scale management API endpoints

The following topics describe the IBM Spectrum Scale management API endpoints in detail.

For more information about using the REST API endpoints, see *IBM Spectrum Scale management API* in *IBM Spectrum Scale: Administration Guide*.

Access/acls: GET

Gets the list of all REST API access control lists (ACL).

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET access/acls` request gets the detailed information of all the REST API access control lists.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/access/acls
```

where

access/acls
: Displays all the REST API acls. Required.

Request headers

```
Accept: application/json
```

Parameters

No parameters.

Request data

No request data.

Response data

```
{"status": {
    "code": Return Code,
    "message": "Return Message"
  },
  "acls": [
    {
      "userGroup": "user group name",
      "entries": [
        {
          "entryId": The access control entry ID,
          "type": "ALLOW| DENY",
          "method": "GET | POST | DELETE | PUT ",
          "uri": "Resource URL"
        }
      ]
    }
  ]
}
```

The details of the parameters are provided in the following list:

status:
: Return status.

  "code": *ReturnCode*,
  : The HTTPS Status code.

"message": "*ReturnMessage*"
The detailed success or error message.

acls:
An array of information on the access control lists.

userGroup:
The user group for which the access control list is defined.

entries
The access control entries.

entryId
The REST API access control list entry ID.

type
The type of access provided. For example, *Allow* | *Deny*

method
The REST API request method which are *GET* | *PUT* | *POST* | *DELETE*

uri
The resource URL.

Examples

The following example gets information all the REST API ACLs.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/access/acls'
```

Response data.

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "acls" : [ {
    "userGroup" : "Administrator",
    "entries" : [ {
      "entryId" : 1,
      "type" : "ALLOW",
      "method" : "GET",
      "uri" : "*"
    }, {
      "entryId" : 2,
      "type" : "ALLOW",
      "method" : "POST",
      "uri" : "/filesystems/*"
    } ]
  }, {
    "userGroup" : "ProtocolAdmin",
    "entries" : [ {
      "entryId" : 1,
      "type" : "ALLOW",
      "method" : "GET",
      "uri" : "*"
    }, {
      "entryId" : 2,
      "type" : "ALLOW",
      "method" : "POST",
      "uri" : "/filesystems/*"
    } ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Access/acls/{userGroup}: GET

Gets the list of the REST API access control lists (ACL) defined for a user group.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET access/acls/{userGroup}` request gets the detailed information of the REST API access control lists (ACL) defined for a user group.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/access/acls/{userGroup}
```

where

acls/{userGroup}
List the REST API access control lists (ACL) defined for the specified user group. Required.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 37. List of parameters

Parameter name	Description and applicable keywords	Required/optional
userGroup	The user group name for which the ACL is to be listed.	Required.

Request data

No request data.

Response data

```
{"status": {
    "code": Return Code,
    "message": "Return Message"
  },
  "acls": [
    {
      "userGroup": "user group name",
      "entries": [
        {
          "entryId": The access control entry ID,
          "type": "ALLOW| DENY",
          "method": "GET | POST | DELETE | PUT ",
          "uri": "Resource URL"
        }
      ]
    }
  ]
```

```
}
```

The details of the parameters are provided in the following list:

status:
Return status.

"code": *ReturnCode*,
The HTTPS Status code.

"message": "*ReturnMessage*"
The detailed success or error message.

acls:
An array of information on the access control lists.

userGroup:
The user group for which the access control list is defined.

entries
The access control entries.

entryId
The REST API access control list entry ID.

type
The type of access provided. For example, *Allow* | *Deny*

method
The REST API request method which are *GET* | *PUT* | *POST* | *DELETE*

uri
The resource URL.

Examples

The following example gets the REST API ACLs for the user group `Monitor`.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/access/acls/Monitor'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "acls" : [ {
    "userGroup" : "Monitor",
    "entries" : [ {
      "entryId" : 1,
      "type" : "ALLOW",
      "method" : "DELETE",
      "uri" : "*"
    }, {
      "entryId" : 2,
      "type" : "ALLOW",
      "method" : "GET",
      "uri" : "*"
    } ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Access/acls/{userGroup}: DELETE

Deletes all the REST API access control lists (ACL) that is defined for a user group.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `DELETE access/acls/{userGroup}` command deletes the ACLs defined for the specified user group.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/access/acls/userGroup
```

where:

`acls/`*userGroup*
: Specifies the user group from which the ACLs are to be deleted. Required.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 38. List of parameters

Parameter name	Description and applicable keywords	Required/optional
userGroup	The user group name from which the ACLs are to be deleted.	Required.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"command names",
             "progress":"progress information,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"Messages",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
```

```
          "runtime":Time",
          "status":"RUNNING | CANCELLING | CANCELLED |COMPLETED | FAILED",
          "pids":"Process IDs
           }
        ],
    }
```

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*command names*'
Array of commands that are run in this job.

"progress":"*information*'
Progress information for the request.

"exitCode":"*Exit code*"
The Exit code of command. The value zero denotes success and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from. *stderr.*

"stdout":"*message*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":"*job data* "
The data that was posted when the job was submitted.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"runtime":*Time*
The runtime of the job.

"status":"RUNNING | COMPLETED | FAILED| CANCELLED| CANCELLING"
Status of the job.

"pids": "Process IDs"
The process IDs of all the active sub processes that are started to manage the job.

Examples

The following example deletes the ACL entries from the user group `Protocol Admin`.

Request data:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https:///198.51.100.1:443/scalemgmt/v2/access/acls/ProtocolAdmin'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000010,
    "status" : "COMPLETED",
    "submitted" : "2021-01-30 17:10:31,554",
    "completed" : "2021-01-30 17:10:32,787",
    "runtime" : 1233,
    "request" : {
      "type" : "DELETE",
      "url" : "/scalemgmt/v2/access/acls/ProtocolAdmin"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mv --force '/var/lib/mmfs/gui/settings.json.77cf20b5-b767-4997-ac98-
cefe18e2d93c' '/var/lib/mmfs/gui/settings.json' " ],
      "stdout" : [ "info: chrestacl ProtocolAdmin remove --force --verbose " ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Access/acls/{userGroup}/entry/{entryId}: DELETE

Deletes a specified REST API access control lists (ACL) entry that is defined for a user group.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `DELETE access/acls/{userGroup}/entry/{entryID}` command deletes the specified ACL entry that is defined for a user group.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/access/acls/userGroup/entry/
entryId
```

where:

acls/userGroup
Specifies the user group with which the ACL entries are associated. Required.

entry/*entryId*
Specifies the ACL entry ID that needs to be deleted. Required.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 39. List of parameters

Parameter name	Description and applicable keywords	Required/optional
userGroup	The name of the user group for which the ACL entries are defined.	Required.
entryID	The ID of the ACL entry that is to be deleted.	Required.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"command names",
             "progress":"progress information,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"Messages",
           },
```

```
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted":"Time",
        "completed":Time",
        "runtime":Time",
        "status":"RUNNING | CANCELLING | CANCELLED |COMPLETED | FAILED",
        "pids":"Process IDs
         }
       ],
 }
```

The details of the parameters are provided in the following list:

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*command names*'
Array of commands that are run in this job.

"progress":"*information*'
Progress information for the request.

"exitCode":"*Exit code*"
The Exit code of command. The value zero denotes success and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from. *stderr.*

"stdout":"*message*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":"*job data* "
The data that was posted when the job was submitted.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"runtime":*Time*
The runtime of the job in milliseconds.

"status":"RUNNING | COMPLETED | FAILED| CANCELLING |CANCELLED"
Status of the job.

"pids": "Process IDs"
The process IDs of all the active sub processes that are started to manage the job.

Examples

The following example deletes the ACL entry with the ID 1 from the user group `CSI Admin`.

Request data:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https:///198.51.100.1:443/scalemgmt/v2/access/acls/CsiAdmin/entry/1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000017,
    "status" : "COMPLETED",
    "submitted" : "2021-01-30 18:29:58,512",
    "completed" : "2021-01-30 18:29:59,026",
    "runtime" : 514,
    "request" : {
      "type" : "DELETE",
      "url" : "/scalemgmt/v2/access/acls/CsiAdmin/entry/1"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mv --force '/var/lib/mmfs/gui/settings.json.6d00b312-8bba-4645-
bee8-0996b4f3dbc6' '/var/lib/mmfs/gui/settings.json' " ],
      "stdout" : [ "info: chrestacl CsiAdmin remove --entryId 1 --force --verbose " ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Access/acls/{userGroup}: POST

Copies the REST API ACL entries from an existing user group.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `access/acls/{userGroup}` request copies all REST API ACL entries from an existing user group and overwrites the ones that are defined for the specified group.

Request URL

```
https://<IP address or host name of API server>:port/scalemgmt/v2/access/acls/userGroup
```

where:

acls/ userGroup
: Specifies the user group name where the copied ACL entries are added. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 40. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
userGroup	The name of the user group where the copied entries are added.	Required.
copyFrom	The name of the user group from where the ACL entries are copied.	Optional.

Request data

No request data.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
```

```
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted":"Time",
        "completed":Time",
        "status":"Job status",
        "pids":"Process IDs
        }
 ],
}
```

The details of the parameters are provided in the following list:

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from stderr.

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

"pids":"Process ID"
The process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to copy ACL entries from the user group `Administrator` to the user group `CsiAdmin`.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/access/acls/CsiAdmin?copyFrom=Administrator'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000003,
    "status" : "COMPLETED",
    "submitted" : "2021-01-30 18:01:45,173",
    "completed" : "2021-01-30 18:01:45,737",
    "runtime" : 564,
    "request" : {
      "type" : "POST",
      "url" : "/scalemgmt/v2/access/acls/CsiAdmin"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mv --force '/var/lib/mmfs/gui/settings.json.33cc0d46-
d623-43ca-86a9-508623c85b5c' '/var/lib/mmfs/gui/settings.json' " ],
      "stdout" : [ "info: chrestacl CsiAdmin copy --from Administrator --force --verbose " ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Access/acls/{userGroup}: PUT

Adds a REST API access control list (ACL) entry for a user group.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `access/acls/{userGroup}` request adds a REST API ACL entry for the specified user group.

Note: Only the users with *Security Admin* and *User Admin* roles can add an ACL entry for a user group.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/access/acls/userGroup
```

where

acls/*userGroup*
Specifies the user group for which the ACL entry is added. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 41. List of parameters

Parameter name	Description and applicable keywords	Required/optional
userGroup	The user group name for which the ACL entry is to be created.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
                "type":"{allow | deny}",
                "method":"GET | DELETE | PUT | POST",
                "endpoint":"Endpoint name",
                "allEndpoints":"TRUE | FALSE",
}
```

"type":"allow | deny "
Specifies the permission type of the entry.

"method":"GET | POST | PUT | DELETE"
The HTTP method for which the permission is defined.

"endpoint":"*Endpoint Name* "
The name of the endpoint for which the permission is being defined. It is mutually exclusive with the `allEndpoints` field.

"allEndpoints":"True | False"
Specifies whether the permission is defined for all REST API endpoints. It is mutually exclusive with the `endpoint` field.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         "pids":"Process IDs",
         }
   ],
  }
```

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of the command. The value zero denotes success, and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from stderr.

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":"*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

"pids":"process IDs"
The process IDs of all the active sub processes that manage the job.

Examples

The following example sets ACL information for the endpoint `/filesystems/{filesystemName}/filesets` and the user group `Csi Admin`.

Request data:

```
 "type": "deny",
         "method": "POST",
         "endpoint": "/filesystems/{filesystemName}/filesets",
         "allEndpoints": false
```

Corresponding request URL:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
-d '{
   "type": "deny",
   "method": "POST",
    "endpoint": "/filesystems/{filesystemName}/filesets",
    "allEndpoints": false
     }' 'https://198.51.100.1:443/scalemgmt/v2/acess/acls/CsiAdmin'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000010,
    "status" : "COMPLETED",
    "submitted" : "2021-01-30 18:18:38,755",
    "completed" : "2021-01-30 18:18:39,627",
    "runtime" : 872,
    "request" : {
      "type" : "PUT",
      "url" : "/scalemgmt/v2/access/acls/CsiAdmin"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mv --force '/var/lib/mmfs/gui/settings.json.5d599bb5-51d6-4117-
a115-345f2808145b' '/var/lib/mmfs/gui/settings.json' " ],
      "stdout" : [ "info: chrestacl CsiAdmin add --type deny --method POST --endpoint '/
filesystems/{filesystemName}/filesets' --verbose " ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
```

```
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Bucket/keys: PUT

Sets or changes keys for a bucket.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `bucket/keys` request sets or changes keys for a bucket. For more information about the fields in the data structures that are returned, see “mmafmcoskeys command” on page 58.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/bucket/keys
```

where

bucket/keys
Specifies the target of the request.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 42. List of parameters

Parameter name	Description and applicable keywords	Required/optional
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
  "bucket": "Bucket name",
  "accessKey": "Access key",
  "secretKey": "Secret key"
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"bucket": "*Bucket name*"
Name of the bucket.

"accessKey": "*Access key*"
Access key of the specified bucket.

"secretKey": "*Secret key*"
Secret key of the specified bucket.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to set or change bucket key:

Request data:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json' -d '{ \
   "bucket": "mybucket", \
   "accessKey": "ACCESS_KEY", \
   "secretKey": "SECRET_KEY" \
 }' 'https://198.51.100.1:443/scalemgmt/v2/bucket/keys'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
    "jobs" : [ {
    "jobId" : 1000000000025,
    "status" : "COMPLETED",
    "submitted" : "2020-09-28 14:50:41,911",
    "completed" : "2020-09-28 14:50:42,872",
    "runtime" : 961,
    "request" : {
      "data" : {
        "accessKey" : "****",
        "bucket" : "mybucket",
        "secretKey" : "****"
      },
      "type" : "PUT",
      "url" : "/scalemgmt/v2/bucket/keys"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmafmcoskeys 'mybucket' set  ****  **** " ],
      "stdout" : [ ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Related reference

“mmafmconfig command” on page 45
Can be used to manage home caching behavior and mapping of gateways and home NFS exported servers.

Bucket/keys/{bucketName}: DELETE

Deletes local bucket definition with keys.

Availability

Available on all IBM Spectrum Scale editions.

Description

The DELETE `bucket/keys/`*{bucketName}* request deletes local bucket definition with keys. For more information about the fields in the data structures that are returned, see "mmafmcoskeys command" on page 58.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/bucket/keys/{bucketName}
```

where

`bucket/keys/`*{bucketName}*
: Specifies the bucket to be deleted. Required.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 43. List of parameters

Parameter name	Description and applicable keywords	Required/optional
{bucketName}	Name of the bucket to be deleted.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
```

```
            "exitCode":"Exit code",
            "stderr":"Error",
            "stdout":"String",
          },
        "request":" ",
          {
            "type":"{GET | DELETE | DELETE | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted":"Time",
        "completed":Time",
        "status":"Job status",
        }
      ],
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to delete a local bucket with keys:

Request data:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/bucket/keys/myBucket'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000002,
    "status" : "COMPLETED",
    "submitted" : "2020-09-29 14:34:07,383",
    "completed" : "2020-09-29 14:34:08,296",
    "runtime" : 913,
    "request" : {
      "data" : "myBucket",
      "type" : "DELETE",
      "url" : "/scalemgmt/v2/bucket/keys/myBucket"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmafmcoskeys 'myBucket' delete " ],
      "stdout" : [ ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Related reference

“mmafmcoskeys command” on page 58
Manages an access key and a secret key to access a bucket on a cloud object storage.

CES/addresses: GET

Gets information about CES addresses.

Availability

Available on all IBM Spectrum Scale editions.

Description

The GET `ces/addresses` request gets information about CES (Cluster Export Services) addresses. For more information about the fields in the data structures that are returned, see “mmces command” on page 133.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/ces/addresses
```

where

`ces/addresses`
Specifies CES address as the resource. Required.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 44. List of parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Request data

No request data.

Response data

The following list of attributes are available in the response data:

```
{
"status":
   {
      "code": ReturnCode
      "message": "ReturnMessage",
   }
"paging":
```

```
  {
    "next": "URL"
  },
   "cesaddresses": [
      {
         "oid": "Integer"
         "nodeNumber": "Integer"
         "nodeName": Node name
         "attributes": "Attributes",
         "cesAddress": "IP",
         "cesGroup": "Group",
         }
   ],
}
```

The details of the parameters are provided in the following list:

"status":
Return status.

"code": *ReturnCode*,
The HTTP status code that was returned by the request.

"message": "*ReturnMessage*"
The return message.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"cesaddresses":
An array of information about CES addresses. For more information about the fields in this structure, see the links at the end of this topic.

"oid": "*Integer*"
Internal identifier that is used for paging.

"nodeNumber": "*Integer*"
The number of the cesNode this address belongs to.

"nodeName": "*Node*"
The CES node to which the address is assigned.

"Attributes": "*Attributes*"
Protocol attributes that are associated with the CES address.

"cesAddress": "*IP*"
The CES address that is assigned to the node.

"cesGroup": "Group"
The group to which the CES address is assigned.

Examples

The following example gets information about the CES addresses available in the system:

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/ces/addresses'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

The **cesaddresses** array returns four objects in the following example. Each object contains details about one CES address.

```
{
  "cesaddresses" : [ {
    "cesAddress" : "198.51.100.35",
    "nodeName" : "mari-13.localnet.com"
  }, {
    "cesAddress" : "198.51.100.37",
    "nodeName" : "mari-12.localnet.com"
  }, {
    "cesAddress" : "198.51.100.31",
    "nodeName" : "mari-12.localnet.com"
  }, {
    "cesAddress" : "198.51.100.33",
    "nodeName" : "mari-14.localnet.com"
  }, {
    "cesAddress" : "198.51.100.27",
    "nodeName" : "mari-14.localnet.com"
  }
  ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully"
  }
}
```

Using the field parameter ":all:" returns entire details of the CES addresses available in the system:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/ces/addresses?fields=:all:'
```

Response data:

```
{
  "cesaddresses" : [ {
    "attributes" : "",
    "cesAddress" : "198.51.100.35",
    "cesGroup" : "",
    "nodeName" : "mari-13.localnet.com",
    "nodeNumber" : 3,
    "oid" : 1
  }, {
    "attributes" : "",
    "cesAddress" : "198.51.100.37",
    "cesGroup" : "",
    "nodeName" : "mari-12.localnet.com",
    "nodeNumber" : 2,
    "oid" : 2
  }, {
    "attributes" : "",
    "cesAddress" : "198.51.100.31",
    "cesGroup" : "",
    "nodeName" : "mari-12.localnet.com",
    "nodeNumber" : 2,
    "oid" : 3
  }, {
    "attributes" : "",
    "cesAddress" : "198.51.100.33",
    "cesGroup" : "",
    "nodeName" : "mari-14.localnet.com",
    "nodeNumber" : 4,
    "oid" : 4
  }, {
    "attributes" : "",
    "cesAddress" : "198.51.100.27",
    "cesGroup" : "",
    "nodeName" : "mari-14.localnet.com",
    "nodeNumber" : 4,
    "oid" : 5
  }
  ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully"
  }
}
```

Related reference

“mmces command” on page 133

Manage CES (Cluster Export Services) configuration.

CES/addresses/{cesAddress}: GET

Gets information about a CES address.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET ces/addresses/{cesAddress}` request gets information about a specific CES (Cluster Export Services) address. For more information about the fields in the data structures that are returned, see "mmces command" on page 133.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/ces/addresses/CesAddress
```

where

ces/addresses
Specifies CES address as the resource. Required.

CesAddress
Specifies the CES address about which you want to get information. Required.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 45. List of parameters

Parameter name	Description and applicable keywords	Required/optional
cesAddress	The IP address to query.	Required.

Request data

No request data.

Response data

The following list of attributes are available in the response data:

```
{
"status":
   {
      "code": ReturnCode
      "message": "ReturnMessage",
   }
"paging":
  {
    "next": "URL"
  },
   "cesaddresses": [
      {
         "oid": "Integer"
```

```
          "nodeNumber": "Integer"
          "nodeName": Node name
          "attributes": "Attributes",
          "cesAddress": "IP",
          "cesGroup": "Group",
          }
   ],
}
```

The details of the parameters are provided in the following list:

"status":
Return status.

"code": *ReturnCode*,
The HTTP status code that was returned by the request.

"message": "*ReturnMessage*"
The return message.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"cesaddresses":
An array of information about CES addresses. For more information about the fields in this structure, see the links at the end of this topic.

"oid": "*Integer*"
Internal identifier that is used for paging.

"nodeNumber": "*Integer*"
The number of the cesNode this address belongs to.

"nodeName": "*Node*"
The CES node to which the address is assigned.

"Attributes": "*Attributes*"
Protocol attributes that are associated with the CES address.

"cesAddress": "*IP*"
The CES address that is assigned to the node.

"cesGroup": "*Group*"
The group to which the CES address is assigned.

Examples

The following example gets information about the CES address `198.51.100.8`.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/ces/addresses/198.51.100.8'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "cesaddresses": [
    {
    "attributes" : "",
    "cesAddress" : "198.51.100.8",
    "cesGroup" : "",
    "nodeName" : "mari-13.localnet.com",
```

```
    "nodeNumber" : 3,
    "oid" : 1
    }
  ]
}
```

Related reference

“mmces command” on page 133
Manage CES (Cluster Export Services) configuration.

CES/services: GET

Gets information about CES services.

Availability

Available on all IBM Spectrum Scale editions.

Description

The GET `ces/services` request gets information about the CES (Cluster Export Services) services in the cluster. For more information about the fields in the returned data structure, see “mmces command” on page 133.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/ces/services
```

where

ces/services
: Specifies CES services as the resource. Required.

Request headers

```
Accept: application/json
```

Parameters

No parameters.

Request data

No request data.

Response data

```
{
   "status": {
      "code": ReturnCode,
      "message": "ReturnMessage"
   }
   "cesservices": {
        "protocolStates": [
         {
            "service":"Service"
            "enabled":"{yes | no}",
         }
       ]
      "protocolNodes": [
         {
            "nodeName":"Node"
            "serviceStates": [
               {
                  "service":"Service"
                  "running":"{yes | no}",
               }
            ]
         }
      ],
   },

}
```

The details of the parameters are provided in the following list:

status:
Return status.

"code": *ReturnCode*,
The return message.

"message": "*ReturnMessage*"
The return message.

cesservices:
Information about CES services. The information consists of two arrays, `protocolNodes` and `protocolStates`. Each array contains multiple elements, with one element for each CES service. For more information about the fields in these structures, see the link at the end of this topic.

protocolStates:
An array of information about the services. Each element describes one service:

"service":"*Service*"
Identifies the service such as CES, NETWORK, NFS, or SMB.

"enabled":"{yes | no}"
Indicates whether the service is enabled.

protocolNodes
An array of information about protocol nodes. Each array element describes one protocol node.

"nodeName":"*Node*"
The name of the node.

serviceStates:
An array of information about the services for which the node is a protocol node.

"service":"*Service*"
Name of the service.

"running":"{yes | no}"
Indicates whether the service is running.

Examples

The following example gets information about the CES services.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/ces/services'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

The **protocolNodes** array returns the details of the protocols configured in each protocol node. The **protocolStates** array provides the status of the protocols.

```
{
  "cesservices" : {
    "protocolNodes" : [ {
      "nodeName" : "mari-12.localnet.com",
      "serviceStates" : [ {
        "running" : "yes",
        "service" : "OBJ"
      }, {
        "running" : "yes",
        "service" : "SMB"
      }, {
        "running" : "yes",
        "service" : "NFS"
```

```
      } ]
    }, {
      "nodeName" : "mari-13.localnet.com",
      "serviceStates" : [ {
        "running" : "yes",
        "service" : "OBJ"
      }, {
        "running" : "yes",
        "service" : "SMB"
      }, {
        "running" : "yes",
        "service" : "NFS"
      } ]
    }, {
      "nodeName" : "mari-14.localnet.com",
      "serviceStates" : [ {
        "running" : "yes",
        "service" : "OBJ"
      }, {
        "running" : "yes",
        "service" : "SMB"
      }, {
        "running" : "yes",
        "service" : "NFS"
      } ]
    }, {
      "nodeName" : "mari-15.localnet.com",
      "serviceStates" : [ {
        "running" : "yes",
        "service" : "OBJ"
      }, {
        "running" : "yes",
        "service" : "SMB"
      }, {
        "running" : "yes",
        "service" : "NFS"
      } ]
    } ],
    "protocolStates" : [ {
      "enabled" : "yes",
      "service" : "OBJ"
    }, {
      "enabled" : "no",
      "service" : "BLOCK"
    }, {
      "enabled" : "yes",
      "service" : "SMB"
    }, {
      "enabled" : "yes",
      "service" : "NFS"
    } ]
  },
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully"
  }
}
```


Related reference

“mmces command” on page 133
Manage CES (Cluster Export Services) configuration.

CES/services/{service}: GET

Gets information about a CES service.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET ces/services/service` request gets information about a CES (Cluster Export Services) service in the cluster. For more information about the fields in the returned data structure, see "mmces command" on page 133.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/ces/services/service
```

where

ces/services
Specifies ces/services as the resource.

service
Specifies the service about which you want to get information. Required.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 46. List of parameters

Parameter name	Description and applicable keywords	Required/optional
service	Name of the service about which you need the details.	Required.

Request data

No request data.

Response data

```
{
   "status": {
      "code": ReturnCode,
      "message": "ReturnMessage"
   }
   "cesservices": {
        "protocolStates": [
         {
            "service":"Service"
            "enabled":"{yes | no}",
         }
       ]
      "protocolNodes": [
```

```
          {
             "nodeName":"Node"
             "serviceStates": [
                {
                   "service":"Service"
                   "running":"{yes | no}",
                }
             ]
          }
       ],
    },

}
```

status:
Return status.

"code": *ReturnCode*,
The return message.

"message": "*ReturnMessage*"
The return message.

cesservices:
Information about CES services. The information consists of two arrays, `protocolNodes` and `protocolStates`. Each array contains multiple elements, with one element for each CES service. For more information about the fields in these structures, see the link at the end of this topic.

protocolStates:
An array of information about the services. Each element describes one service:

"service":"*Service*"
Identifies the service such as CES, NETWORK, NFS, or SMB.

"enabled":"{yes | no}"
Indicates whether the service is enabled.

protocolNodes
An array of information about protocol nodes. Each array element describes one protocol node.

"nodeName":"*Node*"
The name of the node.

serviceStates:
An array of information about the services for which the node is a protocol node.

"service":"*Service*"
Name of the service.

"running":"{yes | no}"
Indicates whether the service is running.

Examples

The following example gets information about the SMB service.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/ces/services/SMB'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

The **protocolNodes** array returns the details of the node on which the SMB protocol is configured. The **protocolStates** array provides the status of the SMB protocol.

```
{
  "status": {
    "code": "200",
    "message": "The request finished successfully"
  },
  "cesservices": {
    "protocolStates": [
      {
        "service": "SMB",
        "enabled": "yes"
      }
    ],
    "protocolNodes": [
      {
        "nodeName": "testnode-1.localnet.com",
        "serviceStates": [
          {
            "service": "SMB",
            "running": "yes"
          }
        ]
      }
    ]
  }
}
```

Related reference

“mmces command” on page 133
Manage CES (Cluster Export Services) configuration.

Cliauditlog: GET

Gets the list of GPFS audit events.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET cliauditlog` request gets a record of various actions that are performed in the system. This helps the system administrator to audit the commands and tasks the users and administrators are performing. These logs can also be used to troubleshoot issues that are reported in the system. For more information about the fields in the data structures that are returned, see “mmaudit command” on page 92.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/cliauditlog
```

where

cliauditlog
: Specifies that the GET request fetches the audit details of the various actions that are performed in the cluster.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 47. List of parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Request data

No request data.

Response data

The following list of attributes are available in the response data:

```
{
"status":
```

```
   {
      "code": ReturnCode
      "message": "ReturnMessage",
   },
"paging":
   {
      "next": Next page URL
      "fields": "Fields",
      "filter": Filter
      "baseUrl": "URL",
      "lastID": ID
      ,
   },
{
   "auditLogRecords": [
      {
         "oid": "ID"
         "arguments": "Arguments"
         "command": "Commands"
         "node": "Nodes"
         "returnCode": "Return code"
         "originator": "Originator"
         "user": "User"
         "pid": "PID"
         "entryTime": "Entry time"
         "exitTime": "Exit time"

       }
}
```

"status":
Return status.

"code": *ReturnCode*,
The HTTP status code that was returned by the request.

"message": "*ReturnMessage*"
The return message.

"paging":
An array of information about the paging information that is used for displaying the details.

"next": "*Next page URL*"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects would be returned by the query.

"fields": "*Fields in the request*"
The fields that are used in the original request.

"filter": "*Filters used in the request*"
The filter that is used in the original request.

"baseUrl": "*URL*"
The URL of the request without any parameters.

"lastId": "*ID*"
The ID of the last element that can be used to retrieve the next elements.

"auditLogRecords":
An array of information that provides the action that is performed.

"oid": "*ID*"
ID used for paging.

"arguments": "*Arguments*"
Arguments of a GPFS command.

"command": "*Command name*"
Name of the GPFS command.

"node": "*Node*"
Name of the node where a GPFS command was running.

"returnCode": "*Return code*"
Return Code of the GPFS command.

"originator": "*Originator*"
Originator of the GPFS command.

"user": "*User name*"
The user who issued the GPFS command.

"pid": "*ID*"
PID of the GPFS command.

"entryTime": "*Entry time*"
Entry time of the GPFS command.

"exitTime": "*Exit time*"
Exit time of the GPFS command.

The return information and the information that the command retrieves are returned in the same way as they are for the other requests. The parameters that are returned are the same as the configuration attributes that are displayed by the `mmaudit` command. For more information, see the "mmaudit command" on page 92.

Examples

The following example gets information about the cluster configuration.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/cliauditlog'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

The **cliauditlog** array provides information about the action that is performed on the cluster.

```
{
  "status": {
    "code": 200,
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/filesystems/gpfs0/filesets?lastId=10001",
    "fields": "period,restrict,sensorName",
    "filter": "usedInodes>100,maxInodes>1024",
    "baseUrl": "/scalemgmt/v2/perfmon/sensor/config",
    "lastId": 10001
  },
  "auditLogRecords": [
    {
      "oid": 0,
      "arguments": "-A yes -D nfs4 -k nfs4",
      "command": "mmchfs",
      "node": "testnode-11",
      "returnCode": 0,
      "originator": "GUI",
      "user": "root",
      "pid": 7891,
      "entryTime": "2017-07-30 14:00:00",
      "exitTime": "2017-07-31 18:00:00"
    }
  ]
}
```

"mmaudit command" on page 92
Manages setting and viewing the file audit logging configuration in IBM Spectrum Scale.

Cluster: GET

Gets details of the cluster.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET cluster` request gets information about the local cluster. For more information about the fields in the data structures that are returned, see the following topics: “mmlscluster command” on page 492, “mmchfs command” on page 232, and “mmlsfs command” on page 506.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/cluster
```

where

cluster
Specifies the IBM Spectrum Scale cluster as the resource of the GET call.

Request headers

```
Accept: application/json
```

Request data

No request data.

Response data

```
{
  "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   }
   "cluster": {
       "clusterSummary": {
         "clusterName": "ClusterName",
         "clusterId": ID,
         "uidDomain": "Domain"
         "rshPath": "RemoteShellCommand",
         "rshSudoWrapper": "Path",
         "rcpPath": "RemoteFileCopyCommand",
         "rcpSudoWrapper": "Path",
         "repositoryType": "CCR | non-CCR",
         "primaryServer": "Server",
         "secondaryServer": "Server",
      }
        "cnfsSummary": {
         "cnfsSharedRoot": "Directory"
         "cnfsMoundtPort": "Port",
         "cnfsNFSDprocs": "Number",
         "cnfsReboot": "{true | false}",
         "cnfsMonitorEnabled": "enabled | disabled",
         "cnfsGanesha": "ServerAddress",
      }
       "cesSummary": {
         "cesSharedRoot": "Directory",
         "enabledServices": "ServiceList",
         "logLevel": "Level",
         "addressPolicy": "{none | balanced-load | node-affinity | even-coverage}",
          }
       "capacityLicensing": {
```

```
        "clusterId": {
             "clusterID": "ID",
              },
        "liableNsdCount": "Number",
        "liableCapacity": "Capacity",
        "liableNsds": [
              {
             "clusterID": "ID",
              },
        "nsdName": "Number",
        "liableCapacity": "Capacity",
        "lastUpdate": "Details,
               {
           "date": "Date",
           "dayAsString": "Day",
           "monthAsString": "Month",
           "fsccDate": "Date",
           "timeAsAmericanString": "Time in US format",
           "isoDateTimeString": "Date in standard format",
           "isoDateTimeMillisString": "Time and date",
           "usDateTimeString": "Date and time",
           "hourMinSecAsString": "Hour, minute, and seconds",
           "timeAsString": "Time",
           "dateAsEnglishString": "Date",
           "isodateAsString": "Date",
           "dateForCnbackup": "Date",
           "yearAsString": "Year",
           "null": "{true | false}",
           "singleton": "{true | false}"
         },
          "uniqueID": "ID"
       },
     ],
  },
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

Note: The structures `cesNode`, `cesSummary`, `cnfsNode`, `cnfsSummary`, and `gatewayNode` appear in the output only when the corresponding role is active on the node.

"cluster":
A data structure that describes the local cluster. It contains the following data structures: `cesSummary`, `clusterSummary`, `cnfsSummary`, `links`, and `nodes`.

"clusterSummary":
Cluster information.

"clusterId": "*ID*"
The ID of the cluster.

"clusterName": "*ClusterName*"
The name of the cluster.

"primaryServer": "*Server*"
The primary server node for GPFS cluster data.

"rcpPath": "*RemoteFileCopyCommand*"
The remote file copy command.

"rcpSudoWrapper": ""
The fully qualified path of the remote file copy program for sudo wrappers.

"repositoryType": "CCR | non-CCR"
The type of repository that the cluster uses for storing configuration data.

"rshPath": "*RemoteShellCommand*"
The remote shell command.

"rshSudoWrapper": "*Path*"
The fully qualified path of the remote shell program for sudo wrappers.

"secondaryServer": "*Server*"
The secondary server node for GPFS cluster data.

"uidDomain": "*Domain*"
The UID domain name of the cluster.

"cnfsSummary":
CNFS cluster information.

"cnfsGanesha" : "*IPAddress*"
The server address of the Ganesha CNFS server.

"cnfsMonitorEnabled" : "enabled" | "disabled"
The state of CNFS monitoring.

"cnfsMoundtPort" : "*Port*"
The port number of the rpc.mountd daemon.

"cnfsNFSDprocs" : "*Number*"
The number of nfsd server threads.

"cnfsReboot" : "true" | "false"
Whether the node restarts when CNFS monitoring detects an unrecoverable problem.

"cnfsSharedRoot" : "Path"
A directory that the CNFS subsystem uses.

"cesSummary":
CES cluster information.

"enabledServices": "*ServiceList*"
A comma-separated list of the services that are enabled. The list can include NFS, SMB, OBJ, BLOCK, or None.

"addressPolicy": "{none | balanced-load | node-affinity | even-coverage}"
The address policy for distributing CES addresses.

"cesSharedRoot": "*Directory*"
The CES shared root directory, which is used for storing CES shared configuration data.

"logLevel": *Level*
The CES log level.

" "capacityLicensing":
Details of capacity for licensing.

"clusterId"

clusterID": "*ID*"
Unique cluster ID.

"nsdName": "*NSD name*",
The name of the NSD.

"liableCapacity": "*Capacity*",
The capacity that is required.

"lastUpdate": "*Details*,

"date": "*Date*",
Date

"dayAsString": "*Day*",
Day

"monthAsString": "*Month*",
Month

"fsccDate": "*Date*",
FSCC date

"timeAsAmericanString": "*Time in US format*",
Time in US format.

"isoDateTimeString": "*Date in standard format*",
Date in standard format

"isoDateTimeMillisString": "*Time and date*",
Time and date

"usDateTimeString": "*Date and time*",
Date and time

"hourMinSecAsString": "*Hour, minute, and seconds*",
Hour, minute, and seconds.

"timeAsString": "*Time*",
Time.

"dateAsEnglishString": "*Date*",
Date.

"isodateAsString": "*Date*",
Date.

"dateForCnbackup": "*Date*",
Date for Cnbackup

"yearAsString": "*Year*",
Year

"null": "{true | false}",
Whether the value is null.

"singleton": "{true | false}"
Whether it is singleton.

"uniqueID": "*ID*"
ID.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example gets information about the local cluster.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/cluster'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "cluster": {
    "clusterSummary": {
      "clusterName": "gpfs-cluster-2.localnet.com",
      "clusterId": "13445038716632536363",
      "uidDomain": "localnet.com",
      "rshPath": "/usr/bin/ssh",
      "rshSudoWrapper": "no",
      "rcpPath": "/usr/bin/scp",
      "rcpSudoWrapper": "no",
      "repositoryType": "CCR",
```

```
      "primaryServer": "testnode-1.localnet.com",
      "secondaryServer": "testnode-2.localnet.com"
    },
    "cnfsSummary": {
      "cnfsSharedRoot": "/mnt/objfs",
      "cnfsMountdPort": "111",
      "cnfsNFSDprocs": "32",
      "cnfsReboot": "yes",
      "cnfsMonitorEnabled": "yes",
      "cnfsGanesha": "198.51.100.8"
    },
    "cesSummary": {
      "cesSharedRoot": "/mnt/objfs/ces",
      "enabledServices": "SMB,NFS",
      "logLevel": "0",
      "addressPolicy": "even-coverage"
    }
"capacityLicensing": {
      "clusterId": {
        "clusterID": "string"
      },
      "liableNsdCount": 26,
      "liableCapacity": 2161114103556,
      "liableNsds": [
        {
          "clusterId": {
            "clusterID": "string"
          },
          "nsdName": "disk37",
          "liableCapacity": 2623752159,
          "lastUpdate": {
            "date": "2019-02-28T06:00:32.539Z",
            "dayAsString": "string",
            "monthAsString": "string",
            "fsccDate": 0,
            "timeAsAmericanString": "string",
            "isoDateTimeString": "string",
            "isoDateTimeMillisString": "string",
            "usDateTimeString": "string",
            "hourMinSecAsString": "string",
            "timeAsString": "string",
            "dateAsEnglishString": "string",
            "isodateAsString": "string",
            "dateForCnbackup": "string",
            "yearAsString": "string",
            "null": true,
            "singleton": true
          },
          "uniqueID": "strin
        }
      ],
      "lastUpdate": {
        "date": "2019-02-28T06:00:32.539Z",
        "dayAsString": "string",
        "monthAsString": "string",
        "fsccDate": 0,
        "timeAsAmericanString": "string",
        "isoDateTimeString": "string",
        "isoDateTimeMillisString": "string",
        "usDateTimeString": "string",
        "hourMinSecAsString": "string",
        "timeAsString": "string",
        "dateAsEnglishString": "string",
        "isodateAsString": "string",
        "dateForCnbackup": "string",
        "yearAsString": "string",
        "null": true,
        "singleton": true
      },
      "uniqueID": "string"
  }
}
```

“mmlscluster command” on page 492
Displays the current configuration information for a GPFS cluster.

“mmchfs command” on page 232
Changes the attributes of a GPFS file system.

“mmlsfs command” on page 506

Displays file system attributes.

Config: GET

Gets information about the configuration of the IBM Spectrum Scale cluster.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET config` request gets information about the local cluster. For more information about the fields in the data structures that are returned, see the following topics: "mmchconfig command" on page 170 and "mmlsconfig command" on page 495.

This API call gets it data from the **mmlsconfig** command. The **mmlsconfig** command displays the PERSISTED configuration values; not the ACTIVE values. That is, the configuration changes made through the **mmchconfig** command comes in to the effect based on how you use **-I** and **-i** parameters of the command. The following three scenarios are applicable:

- **mmchconfig -I**: A value is changed immediately in the daemon but not persistent so that it is reset after a restart. Those settings do not show in **mmlsconfig** at all. In this case, this API call shows an old setting that is temporarily changed to something else.
- **mmchconfig -i**: The setting takes effect immediately and is also persisted so that **mmlsconfig** shows it.
- **mmchconfig**: The setting takes effect on the next restart and is also persisted so that **mmlsconfig** shows it. In this case, the API call shows a future setting that will get applied after the restart.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/config
```

where

config
: Specifies that the GET request fetches the configuration details of the cluster.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 48. List of parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.

Request data

No request data.

Response data

The following list of attributes are available in the response data:

```
{
"status":
   {
      "code": ReturnCode
      "message": "ReturnMessage",
   }
   "config": [
      {
         "clusterConfig": "Configuration details"

       }
}
```

"status":
Return status.

"code": *ReturnCode*,
The HTTP status code that was returned by the request.

"message": "*ReturnMessage*"
The return message.

"config":
An array of information about the cluster configuration.

"clusterConfig": "*Configuration details*"
The cluster configuration as shown by **mmlsconfig -Y** command.

The return information and the information that the command retrieves are returned in the same way as they are for the other requests. The parameters that are returned are the same as the configuration attributes that are displayed by the `mmlsconfig` command. For more information, see the "mmlsconfig command" on page 495.

Examples

The following example gets information about the cluster configuration.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/config'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

The **config** array provides information about the cluster configuration.

```
{
  "config" : {
    "clusterConfig" : {
      "FIPS1402mode" : "no",
      "adminMode" : "central",
      "afmAsyncDelay" : "15",
      "afmAsyncOpWaitTimeout" : "120",
      "afmAtimeXattr" : "no",
      "afmDIO" : "0",
      "afmDirLookupRefreshInterval" : "60",
      "afmDirOpenRefreshInterval" : "60",
      "afmDisconnectTimeout" : "60",
      "afmEnableADR" : "no",
      "afmExpirationTimeout" : "disable",
      "afmFileLookupRefreshInterval" : "30",
      "afmFileOpenRefreshInterval" : "30",
      "afmFlushThreadDelay" : "5",
      "afmGatewayQueueTransfer" : "yes",
```

```
      "afmHardMemThreshold" : "5368709120",
      "afmMaxThrottle" : "0",
      "afmMaxWriteMergeLen" : "1073741824",
      "afmMode" : "read-only",
      "afmMountRetryInterval" : "300",
      "afmNumIOFlushThreads" : "32",
      "afmNumReadThreads" : "1",
      "afmNumWriteThreads" : "1",
      "afmParallelReadChunkSize" : "134217728",
      "afmParallelReadThreshold" : "1024",
      "afmParallelWriteChunkSize" : "134217728",
      "afmParallelWriteThreshold" : "1024",
      "afmRPO" : "disable",
      "afmReadBufferSize" : "33554432",
      "afmReadFlags" : "cached",
      "afmReadSparseThreshold" : "134217728",
      "afmShowHomeSnapshot" : "no",
      "afmSkipWrites" : "(undefined)",
      "afmSyncOpWaitTimeout" : "60",
      "afmWriteBufferSize" : "2097152",
      "atimeDeferredSeconds" : "86400",
      "autoload" : "yes",
      "automountDir" : "/gpfs/automountdir",
      "ccrEnabled" : "yes",
      "cesSharedRoot" : "/mnt/objfs/ces",
      "cifsBypassShareLocksOnRename" : "yes",
      "cifsBypassTraversalChecking" : "yes",
      "cipherList" : "AUTHONLY",
      "clusterId" : "317908494281353658",
      "clusterName" : "gpfsgui-cluster-1.localnet.com",
      "cnfsMountdPort" : "(undefined)",
      "cnfsNFSDprocs" : "32",
      "cnfsReboot" : "yes",
      "cnfsSharedRoot" : "(undefined)",
      "commandAudit" : "syslogOnly",
      "dataCollectionPendingDelay" : "10",
      "dataDiskCacheProtectionMethod" : "0",
      "dataDiskWaitTimeForRecovery" : "3600",
      "dataStructureDump" : "/tmp/mmfs",
      "deadlockBreakupDelay" : "0",
      "deadlockDataCollectionDailyLimit" : "3",
      "deadlockDataCollectionMinInterval" : "3600",
      "deadlockDetectionThreshold" : "300",
      "deadlockDetectionThresholdForShortWaiters" : "60",
      "deadlockDetectionThresholdIfOverloaded" : "1800",
      "deadlockOverloadThreshold" : "1",
      "debugDataControl" : "light",
      "debugDataControlOnSignal" : "heavy",
      "defaultHelperNodes" : "(undefined)",
      "defaultMountDir" : "/gpfs",
      "disableInodeUpdateOnFdatasync" : "no",
      "dmapiDataEventRetry" : "2",
      "dmapiEventTimeout" : "86400000",
      "dmapiFileHandleSize" : "32",
      "dmapiMountEvent" : "all",
      "dmapiMountTimeout" : "60",
      "dmapiSessionFailureTimeout" : "0",
      "enableIPv6" : "no",
      "enforceFilesetQuotaOnRoot" : "no",
      "expelDataCollectionDailyLimit" : "3",
      "expelDataCollectionMinInterval" : "3600",
      "failureDetectionTime" : "35",
      "fastestPolicyCmpThreshold" : "50",
      "fastestPolicyMaxValidPeriod" : "600",
      "fastestPolicyMinDiffPercent" : "50",
      "fastestPolicyNumReadSamples" : "5",
      "fileHeatLossPercent" : "10",
      "fileHeatPeriodMinutes" : "0",
      "indefiniteRetentionProtection" : "no",
      "indirectBlocksPerLockRange" : "10",
      "maxDownDisksForRecovery" : "16",
      "maxFailedNodesForRecovery" : "3",
      "maxFcntlRangesPerFile" : "200",
      "maxFilesToCache" : "4000",
      "maxMBpS" : "2048",
      "maxStatCache" : "1000",
      "maxblocksize" : "16384K",
      "metadataDiskWaitTimeForRecovery" : "2400",
      "minDiskWaitTimeForRecovery" : "1800",
      "minReleaseLevel" : "4.2.3.0",
      "minimumTargetTimeUnderRangeLock" : "30",
      "mmapRangeLock" : "yes",
```

```
      "mmfsLogLevel" : "detail",
      "nistCompliance" : "SP800-131A",
      "noSpaceEventInterval" : "120",
      "nsdRAIDBufferPoolSizePct" : "50",
      "nsdRAIDTracks" : "0",
      "nsdServerWaitTimeForMount" : "300",
      "nsdServerWaitTimeWindowOnMount" : "600",
      "nsdbufspace" : "30",
      "pagepool" : "1G",
      "pagepoolMaxPhysMemPct" : "75",
      "pitWorkerThreadsPerNode" : "0",
      "prefetchThreads" : "72",
      "readReplicaPolicy" : "default",
      "restripeOnDiskFailure" : "no",
      "rpcPerfNumberDayIntervals" : "30",
      "rpcPerfNumberHourIntervals" : "24",
      "rpcPerfNumberMinuteIntervals" : "60",
      "rpcPerfNumberSecondIntervals" : "60",
      "rpcPerfRawExecBufferSize" : "2",
      "rpcPerfRawStatBufferSize" : "6",
      "sidAutoMapRangeLength" : "15000000",
      "sidAutoMapRangeStart" : "15000000",
      "subnets" : "(undefined)",
      "syncSambaMetadataOps" : "yes",
      "systemLogLevel" : "notice",
      "tctEnable" : "yes",
      "tiebreakerDisks" : "(undefined)",
      "tmMaxPhysMemPct" : "75",
      "tscCmdPortRange" : "(undefined)",
      "uidDomain" : "localnet.com",
      "unmountOnDiskFail" : "no",
      "usePersistentReserve" : "no",
      "verbsPorts" : "(undefined)",
      "verbsRdma" : "disable",
      "verbsRdmaCm" : "disable",
      "verbsRdmaPkey" : "32767",
      "verbsRdmaRoCEToS" : "-1",
      "verbsRdmaSend" : "no",
      "verbsRdmasPerConnection" : "16",
      "worker1Threads" : "48"
    }
  },
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully"
  }
}
```


“mmlsconfig command” on page 495

Displays the current configuration data for a GPFS cluster.

“mmchconfig command” on page 170

Changes GPFS configuration parameters.

Diagnostic/snap: GET

Lists the GPFS snap files comprising the diagnostic data that is collected by the **gpfs.snap** command.

Availability

Available on all IBM Spectrum Scale editions.

Description

The GET `diagnostic/snap` request gets a list of the GPFS snap files that are available in the default snap directory. For more information about the fields in the returned data structure, see "gpfs.snap command" on page 8.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/diagnostic/snap
```

where

diagnostic/snap
: Returns the list of GPFS snap files. Required.

Request headers

```
Accept: application/json
```

Parameters

No parameters.

Request data

No request data.

Response data

```
{
  "snaps": [
    {
      "name": "string",
      "size": integer ,
      "timestamp": {
        "date": Time
      },
      "uploaded": true | false
    }
 ],
  "status": {
    "code": ReturnCode,
    "message": "ReturnMessage."
  }
}
```

The details of the parameters are provided in the following list:

snaps
: An array of all the snap files that are taken using either the GUI or the REST API.

name
: Name of the .tar file that comprises the snap files.

size
Size of the .tar file.

timestamp
The time when the snap file was created.

uploaded
Indicates whether the snap has been successfully uploaded to the PMR.

status
The HTTP status code.

message
The detailed success or error message.

Examples

The following example lists the GPFS snap files.

Request data:

```
curl -k -u user:password -X GET --header 'accept:application/json' --header 'Authorization:
Basic YWRtaW46YWRtaW4wMDE=' 'https://198.51.100.1:443/scalemgmt/v2/diagnostic/snap
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "snaps": [
    {
      "name": "/tmp/gpfs.snapOut/3/all.20210128174912.616684.tar",
      "size": 21913600,
      "timestamp": {
        "date": 1611852706089
      },
      "uploaded": false
    },
    {
      "name": "/tmp/gpfs.snapOut/3/all.20210128174417.604073.tar",
      "size": 20203520,
      "timestamp": {
        "date": 1611852410934
      },
      "uploaded": false
    },
    {
      "name": "/tmp/gpfs.snapOut/397370/all.20210128091305.397370.tar",
      "size": 18176000,
      "timestamp": {
        "date": 1611821720755
      },
      "uploaded": false
    },
    {
      "name": "/tmp/gpfs.snapOut/384977/all.20210128090839.384977.tar",
      "size": 18565120,
      "timestamp": {
        "date": 1611821474759
      },
      "uploaded": false
    }
  ],
  "status": {
    "code": 200,
    "message": "The request finished successfully."
  }
}
```

Related reference

“gpfs.snap command” on page 8

Creates an informational system snapshot at a single point in time. This system snapshot consists of information such as cluster configuration, disk configuration, network configuration, network status, GPFS logs, dumps, and traces.

Diagnostic/snap: POST

Creates a new GPFS snap file.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `diagnostic/snap` request creates a new GPFS snap file.

Request URL

```
https://<IP address or host name of API server>:port/scalemgmt/v2/diagnostic/snap
```

where:

`diagnostic/snap`
Creates a new GPFS snap file. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 49. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
 "outputDir": "string",
 "nodesOrNodeClasses": "[Array]",
 "checkSpaceOption": "string",
 "fullCollection":  false,
 "deadlock": false,
 "numberOfDaysBack": backup for number of days,
 "excludeAixDiskAttr": true,
 "excludeAixLvm": true | false,
 "excludeMergeLogs": true | false,
 "excludeNet": true | false,
 "gatherLogs": true |false,
 "mmdf": true |false,
 "performance": true | false,
 "cloudGateway": "string",
 "prefix": "string",
 "protocolTypes": "[]",
 "timeoutInSeconds": Timeout in seconds,
 "purgeFiles": Number of Files purged}
```

"outputDir": "*The output directory*"
The directory path where the output is stored.

"nodesOrNodeClasses": "*Array of nodes*"
List of nodes, node classes, or a mix of both for which the data must be collected.

"checkSpaceOption": "*Type of space check being performed*"
Indicates the manner in which space checking is performed. The values and their interpretations are shown:

- "" - no space check is performed.
- "checkSpace" - space checking is performed before collecting data.
- "checkSpaceOnly" - only space checking is performed but no data is collected.

"fullCollection": "*True | False* "
Indicates whether all large debug files are collected instead of the default behavior of collecting only the new files that were created since a previous run of the **gpfs.snap**.

"deadlock": "*True | False* "
Indicates whether only the minimum amount of data that is necessary to debug a deadlock problem, is collected.

"numberOfDaysBack": "*The number of days of backup*"
Specifies a time limit to reduce the number of large files to be collected.

"excludeAixDiskAttr": "*True | False*"
Indicates whether the AIX disk attributes data will be collected. If *True* the data is not collected.

"excludeAixLvm": "*True | False*"
Indicates whether the AIX Logical Volume Manager (LVM) data is collected. If *True* the data is not collected.

"excludeMergeLogs": "*True | False* "
Indicates whether merge logs and waiters are collected. If *True* the data is not collected.

"excludeNet": "*True | False* "
Indicates whether the network-related information is collected. If *True* the data is not collected.

"gatherLogs": "*True | False* "
Indicates whether all the `mmfs.log` files is gathered, merged, and chronologically sorted.

"mmdf": "*True | False*"
Indicates whether the **mmdf** output is collected.

"performance": "*True | False* "
Indicates whether the performance data is to be gathered.

"cloudGateway": "*The cloud gateway* "
Indicates the manner in which the cloud gateway data is to be gathered. The value and their interpretations are shown:

- NONE - No Transparent cloud tiering data is collected
- BASIC - When the Transparent Cloud Tiering service is enabled, the snap collects information such as logs, traces, Java™ cores, along with minimal system and IBM Spectrum Scale cluster information specific to Transparent Cloud Tiering. No customer sensitive information is collected
- FULL - Additional details such as Java Heap dump are collected, along with the information captured with the BASIC option.

"prefix": "*prefix name* "
Indicates whether the prefix name `gpfs.snap` will be added to the .tar file.

"protocolTypes": "*JSON Array of strings*"
Specifies the type or types of protocol information to be gathered. For example, SMB, NFS, object, authentication, CES, core, or none. By default, whenever any protocol is enabled on a file system, the information is gathered for all types of protocol information.

"timeoutInSeconds": "*time*"
Specifies the timeout value, in seconds, for all commands.

"purgeFiles": "*Number of files*"
Specifies that large debug files will be deleted from the cluster nodes based on the value specified under KeepNumberOfDaysBack parameter. If 0 is specified, all the large debug files are deleted. If a value greater than 0 is specified, large debug files that are older than the number of days specified are deleted. For example, if the value 2 is specified, all files that are older than the previous two days are deleted.

Response data

```
{
  "jobs" : [ {
    "jobId" : ID,
    "status" : "request status",
    "submitted" : "Date and Time",
    "completed" : "Date and Time",
    "runtime" : Time,
    "request" : {
      "type" : "Request Type",
      "url" : "API endpoint URL"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "gpfs.snap --no-check-space " ],
      "stdout" : [ "Result"],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : ReturnCode,
    "message" : "ReturnMessage"
  }
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"runtime":*Time*
The job run time

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"result"

"commands":"*String*'
Array of commands that are run in this job.

"progress":"*String*'
Progress information for the request.

"stdout":"*String*"
CLI messages from stdout.

"exitCode":"*Exit code*"
Exit code of command. Zero is success and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"pids":*Array*
An array of persistent identifiers.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example shows how to create a new GPFS snap file.

Request data:

```
curl -k -u user:password -X POST --header 'content-type:application/json' --header
'accept:application/json' -d '{ \
   "outputDir": "/tmp/gpfs.snapOut/4", \
   "nodesOrNodeClasses": ["gpfsgui-22.novalocal"], \
   "protocolTypes": ["SMB","NFS"], \
   "timeoutInSeconds": 0 \
}' 'h}'  https://openstackhost:2001/scalemgmt/v2/diagnostic/snap
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000001,
    "status" : "COMPLETED",
    "submitted" : "2021-01-22 12:11:45,165",
    "completed" : "2021-01-22 12:18:16,253",
    "runtime" : 391088,
    "request" : {
      "type" : "POST",
      "url" : "/scalemgmt/v2/diagnostic/snap"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "gpfs.snap --no-check-space " ],
      "stdout" : [ "gpfs.snap: started at Fri Jan 22 12:11:45 CET 2021.", "Gathering common
data...", "Gathering Linux specific data...", "Gathering extended network data...", "Gathering
local callhome data...", "Gathering local perfmon data...", "Gathering local msgqueue data...",

"Gathering local sysmon data...", "Gathering local cnss data...", "Gathering local gui
data...", "Gathering trace reports and internal dumps...", "Gathering Transparent Cloud Tiering
data at level BASIC...", "gpfs.snap:  The Transparent Cloud Tiering snap software was not
detected o

n node gpfsgui-11.novalocal", "Gathering QoS data at level FULL...", "gpfs.snap:  No
QoS configuration was found for this cluster.", "gpfs.snap:  QoS configuration collection
complete.", "Gathering cluster wide gui data...", "Gathering cluster wide sysmon data...",
"Gathering cluste

r wide cnss data...", "Gathering cluster wide callhome data...", "Gathering cluster wide
perfmon data...", "Gathering cluster wide msgqueue data...", "gpfs.snap:  Spawning remote
gpfs.snap calls. Master is gpfsgui-11.novalocal.", "This may take a while.", "Copying file /var/
mmfs/tmp

/cleanupAllow/gpfs.snap.gpfsgui-13_20210122121448.2450324.out.tar.gz from
gpfsgui-13.novalocal ...", "Successfully copied file /var/mmfs/tmp/cleanupAllow/
gpfs.snap.gpfsgui-13_20210122121448.2450324.out.tar.gz from gpfsgui-13.novalocal.", "Copying
file /var/mmfs/tmp/cleanupAllow/gpfs
```

```
.snap.gpfsgui-12_20210122121449.3291709.out.tar.gz from
gpfsgui-12.novalocal ...", "Successfully copied file /var/mmfs/tmp/cleanupAllow/
gpfs.snap.gpfsgui-12_20210122121449.3291709.out.tar.gz from gpfsgui-12.novalocal.", "Gathering
cluster wide protocol data...", "Gathering SMB Clust

er wide data...", "Successfully copied file /var/mmfs/tmp/cleanupAllow/
smb.snap.cluster.gpfsgui-12.novalocal.20210122_121729.tar.gz from gpfsgui-12.novalocal.",
"Gathering NFS Cluster wide data...", "Successfully copied file /var/mmfs/tmp/cleanupAllow/
nfs.snap.cluster.gpfsgui-12.nov

alocal.20210122_121730.tar.gz from gpfsgui-12.novalocal.", "Gathering AUTH
Cluster wide data...", "Successfully copied file /var/mmfs/tmp/cleanupAllow/
auth.snap.cluster.gpfsgui-12.novalocal.20210122_121731.tar.gz from gpfsgui-12.novalocal.",
"Gathering CES Cluster wide data...", "Su

ccessfully copied file /var/mmfs/tmp/cleanupAllow/
ces.snap.cluster.gpfsgui-12.novalocal.20210122_121732.tar.gz from gpfsgui-12.novalocal.",
"Gathering waiters from all nodes...", "Gathering mmfs.logs from all nodes.  This may take a
while...", "All data has been collected.  Processi

ng collected data.", "  This may take a while...", "Packaging
master node data...", "Writing * to file /tmp/gpfs.snapOut/1819085/collect/
gpfs.snap.gpfsgui-11_master_20210122121145.1819085.out.tar.gz", "Packaging all data.",
"Writing . to file /tmp/gpfs.snapOut/1819085/all.2021012212

1145.1819085.tar", "/bin/tar: gpfsgui-11_master_20210122121145.1819085/var/log/messages: file
changed as we read it", "gpfs.snap completed at Fri Jan 22 12:18:15 CET
2021", "################################################################################", "Send
file /tmp/gpfs.snapOu

t/1819085/all.20210122121145.1819085.tar to IBM Service", "Examine
previous messages to determine additional required data.",
"################################################################################" ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Diagnostic/snap/{snapPath}: GET

Downloads the specific snap file which comprises the diagnostic data to troubleshoot issues in the system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET diagnostic/snap/{snapPath}` request downloads a specific snap file from the default directory `/tmp/gpfs.snapOut`. For more information about the fields in the returned data structure, see "gpfs.snap command" on page 8.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/diagnostic/snap/{snapPath}
```

where

snap/{snapPath}
Specifies the snap file that is to be downloaded. Required.

Request headers

```
Accept: application/octet-stream
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 50. List of parameters

Parameter name	Description and applicable keywords	Required/optional
snapPath	The snap file path provided as a URL	Required.

Request data

No request data.

Response data

No response data.

Examples

The following example downloads the snap file.

Request data:

```
curl -k -u 'username:password' -X GET --header 'Accept: application/octet-stream' --header
'Authorization: Basic YWRtaW46YWRtaW4wMDE=' 'https://198.51.100.1:443/scalemgmt/v2/diagnostic/
snap/%2Ftmp%2Fgpfs.snapOut%2F355725%2Fall.20210128084441.355725.tar' >/tmp/snap.tar'
```

Response data:

The snap file is downloaded.

Related reference

“gpfs.snap command” on page 8
Creates an informational system snapshot at a single point in time. This system snapshot consists of information such as cluster configuration, disk configuration, network configuration, network status, GPFS logs, dumps, and traces.

Diagnostic/snap/{snapPath}/pmr/{pmrID}: PUT

Uploads a snap file to a PMR. A snap file comprises the diagnostic data that is collected by the **gpfs.snap** command.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `diagnostic/snap/{snapPath}/pmr/{pmrID}` request uploads the snap file to a support case based on a specific PMR ID. For more information about the fields in the data structures that are returned, see "gpfs.snap command" on page 8.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/diagnostic/snap/
{snapPath}/pmr/{pmrID}
```

where

snap/*snapPath*
: The snap file that will be uploaded and its location. Required.

pmr/*pmrID*
: The PMR ID to which the snap file will be associated. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 51. List of parameters

Parameter name	Description and applicable keywords	Required/optional
snapPath	The snap file path	Required.
pmrID	The PMR ID with which the snap file is associated.	Required.

Request data

No request data

Response data

```
{
  "jobs": [
    {
      "jobId": ID,
      "status": "RUNNING| COMPLETED | FAILED",
      "submitted": "Date and Time",
      "completed": "Date and Time",
      "runtime": Time,
```

```
      "request": {
        "type": "PUT |POST|GET|DELETE",
        "url": "URL "
      },
      "result": {},
      "pids": []
    }
  ],
  "status": {
    "code": ReturnCode,
    "message": "Return Message."
  }
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"jobId":"*ID*",
The unique ID of the job.

"status":"RUNNING| COMPLETED | FAILED"
The status of the job

"submitted":"*Time*"
The time at which the job was submitted.

"completed":"*Time*"
The time at which the job was completed.

"runtime":*Time*"
The job runtime

"request"

type:"GET|POST|PUT|DELETE
The HTTP request type

"URL":*URL*
The URL through which the job is submitted

"result"
The request result

"pids
The array of persistent identifiers.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example uploads the snap file to the pmr location /pmr/TS123456789.

Request data:

```
curl -k -u user:password -X PUT --header 'Content-
Type: application/json' --header 'Accept: application/json' --header
'Authorization: Basic YWRtaW46YWRtaW4wMDE=' 'https://198.51.100.1:443/scalemgmt/v2/diagnostic/
snap/%2Ftmp%2Fgpfs.snapOut%2F397370%2Fall.20210128091305.397370.tar/pmr/TS123456789'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs": [
    {
      "jobId": 1000000000001,
      "status": "RUNNING",
      "submitted": "2021-01-29 06:52:29,970",
      "completed": "N/A",
      "runtime": 71,
      "request": {
        "type": "PUT",
        "url": "/scalemgmt/v2/diagnostic/snap/
%2Ftmp%2Fgpfs.snapOut%2F397370%2Fall.20210128091305.397370.tar/pmr/TS123456789"
      },
      "result": {},
      "pids": []
    }
  ],
  "status": {
    "code": 202,
    "message": "The request was accepted for processing."
  }
}
```

Related reference

"gpfs.snap command" on page 8

Creates an informational system snapshot at a single point in time. This system snapshot consists of information such as cluster configuration, disk configuration, network configuration, network status, GPFS logs, dumps, and traces.

Diskmgmt/vdiskset/server/list/{nodeClass}: GET

Gets a list of IBM Spectrum Scale RAID servers and their characteristics.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET scalemgmt/v2/gnr/diskmgmt/vdiskset/server/list/{nodeClass}` request gets the list of IBM Spectrum Scale RAID servers and their characteristics. For more information about the fields in the data structures that are returned, see the **mmvdisk** command in the IBM Spectrum Scale documentation.

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/gnr/diskmgmt/vdiskset/server/list/nodeClass
```

where

list/{nodeClass}
: Specifies the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request data

No request data.

Parameters

Table 52. List of parameters		
Parameter name	**Description and applicable keywords**	**Required/Optional**
nodeClass	The node class or the node name for which details are requested.	Required.

Response data

```
  {
  "status": {
    "code": Status code,
    "message": "Status message"
  },  "stdout": [
     "serverList": [
    {
      "nodeName": "Node name",
      "needsAttention": "string",
      "matchingMetric": "string",
      "diskTopology": "Disk topology",
      "number": Node sequence number
    }
    ],
   ]
}
```

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

"stdout":"*CLI messages*"
The CLI messages from stdout.

"serverList"
An array that comprises details on the jobs that were run.

"nodeName": "*Name of node*"
The name of the node name for which the details are displayed.

"needsAttention": "*yes | no* "
Specifies whether running the **mmvdisk** command faces problems from internal commands like **mmgetpdisktopology** or **tspsummary**.

"matchingMetric":"*Metrics* "
The match between current disk topology and the expected designs.

"disktopology":"*Disk topology*"
The IBM Spectrum Scale RAID disk topology on the recovery group servers.

"nodeNumber":"*Node sequence number*"
The sequence number of the node.

Examples

The following example gets information on the nodeClass *nc1*.

Request data:

```
curl -X GET --header 'Accept: application/json' --header 'Authorization: Basic
YWRtaW46YWRtaW4wMDE='
'https://198.51.100.1:443/scalemgmt/v2/gnr/diskmgmt/vdiskset/server/list/nc1
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.


```
{
  "stdout": [
  "serverList": [
    {
      "diskTopology": "ECE 3 HDD",
      "matchingMetric": "100/100",
      "needsAttention": "no",
      "nodeName": "hciece01-hs.gpfs.net",
      "nodeNumber": 1
    },
    {
      "diskTopology": "ECE 3 HDD",
      "matchingMetric": "100/100",
      "needsAttention": "no",
      "nodeName": "hciece02-hs.gpfs.net",
      "nodeNumber": 2
    },
    {
      "diskTopology": "ECE 3 HDD",
      "matchingMetric": "100/100",
      "needsAttention": "no",
      "nodeName": "hciece03-hs.gpfs.net",
      "nodeNumber": 3
    },
    {
      "diskTopology": "ECE 3 HDD",
      "matchingMetric": "100/100",
      "needsAttention": "no",
      "nodeName": "hciece04-hs.gpfs.net",
      "nodeNumber": 4
    },
    {
      "diskTopology": "ECE 3 HDD",
      "matchingMetric": "100/100",
      "needsAttention": "no",
      "nodeName": "hciece05-hs.gpfs.net",
      "nodeNumber": 5
    }
  ]
  ],
  "status": {
    "code": 200,
    "message": "The request finished successfully."
  }
}
```


Related information

mmvdisk server command in the IBM Spectrum Scale documentation.

Encryption/rkmClients: GET

Gets information on all the clients that belongs to a specified RKM server.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET scalemgmt/v2/encryption/rkmClients` request gets the details for all key clients that are associated with an RKM server. For more information about the fields in the data structures that are returned, see the "mmkeyserv command" on page 469.

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/encryption/rkmClients
```

where

`rkmClients`
Specifies the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Parameters

Table 53. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
keyServer	The name of the server where the client is registered.	Optional
clientName	The name of the registered client.	Optional

Request data

No request data.

Response data

```
{
    "jobs": [
        {
            "jobId": Job ID,
            "status": "Job status ",
            "submitted": "Time and date",
            "completed": "Time and date",
            "runtime": Time,
            "request": {
                "type": "GET | POST | PUT | DELETE",
                "url": "Request URL"
            },
            "result": {
                "progress": [],
                "commands": [
                    "Command name "
```

```
                ],
                "stdout": [
                 {
                   "clientList": [
                  {
                "label": "Client label"
                "keyServer": "Server name"
                "tenantName": "Tenant name"
                "clientName": "Client name"
                "certificateExpiration": "Certificate expiration date"
                "certificateType": "Certificate type"
                }
                ],
                "stderr": [],
                "exitCode": Code            },
            "pids": []
        }
    ],
    "status": {
        "code": Status code,
        "message": "Status message"
    }
}
    }
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"jobId": "*Job ID*"
The unique ID of the job.

"status": "*RUNNING | COMPLETED | FAILED*"
The status of the job.

"submitted": "*Time and date*"
The time and date when the job was submitted.

"completed": "*Time and date*"
The time at which the job was completed.

"runtime": "*Time and date*"
The time that the job took to run.

"request"

"type": "*GET | POST | PUT | DELETE*"
The request type.

"URL": "*Request URL*"
The URL through which the job is submitted.

"result"

"progress": *Job progress*
Progress information for the request.

"commands": "*Command name*
Array of commands that are run in this job.

"stdout": "*message*
Request Information.

"label": "*Client name label*"
The label of the registered client.

"keyServer": "*Server name*"
The name of the server.

"tenantName": "*Tenant name*"
The name of the tenant.

"clientName": "*Client name*"
The name of the registered client.

"certificateExpiration": "*Certificate Expiration date* "
The date and time of certificate expiration.

"certificateType": "*Certificate type*"
The type of certificate.

"exitCode":"*Exit code*"
Exit code of command. Zero indicates success and any other value denotes failure.

"stderr":"*Error*"
CLI messages from stderr.

"pids": "*Process IDs*
The process IDs for the job.

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example gets information on the key clients that are associated with the RKM server.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json' 'https://198.51.100.1:443/
scalemgmt/v2/encryption/rkmClients'?keyServer=lodestar1.fyre.ibm.com&clientName=myClient1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
    "jobs": [
        {
            "jobId": 1000000000001,
            "status": "COMPLETED",
            "submitted": "2021-06-20 09:04:41,521",
            "completed": "2021-06-20 09:04:42,129",
            "runtime": 608,
            "request": {
                "type": "GET",
                "url": "/scalemgmt/v2/encryption/rkmClients"
            },
            "result": {
                "progress": [],
                "commands": [
                    "mmkeyserv client show  --server 'lodestar1.fyre.ibm.com' "
                ],
                "stdout": [
                 {
                 "clientList": [
                 {
                 "certificateExpiration": "2024-06-19 06:45:13 (+0000)",
                 "certificateType": "system-generated",
                 "clientName": "myclient2",
                 "keyServer": "lodestar1.fyre.ibm.com",
                 "label": "myclient2",
                 "tenantNames": "devG1"
                 },
                 {
                 "certificateExpiration": "2021-09-18 10:54:08 (+0000)",
                 "certificateType": "system-generated",
                 "clientName": "myclient3",
                 "keyServer": "lodestar1.fyre.ibm.com",
                 "label": "myclient3",
                 "tenantNames": "(none)"
                 }
                 ],
                 "stderr": [],
                 "exitCode": 0
```

```
            },
            "pids": []
        }
    ],
    "status": {
        "code": 200,
        "message": "The request finished successfully."
    }
}
```

Related reference

“mmkeyserv command” on page 469
Manages encryption key servers and clients.

Encryption/rkmKeys: GET

Displays information about the encryption keys in a tenant.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET scalemgmt/v2/encryption/rkmKeys` request displays information about the encryption keys in a tenant. For more information about the fields in the data structures that are returned, see the "mmkeyserv command" on page 469.

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/encryption/rkmKeys
```

where

rkmKeys
: Specifies the resource of this GET call.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Parameters

Table 54. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
body	Body of the request that contains the parameters that need to be passed on to the IBM Spectrum Scale Erasure Code Edition system to complete the requested operation.	Required

Request data

The following list of attributes is available in the request data:

```
{
  "serverName": "string",
  "tenantName": "string",
  "passwordFile": "string"
}
```

The details of the parameters are given in the following list:

"serverName": *Server name*
: Specifies the host name or IP address of the RKM server to which the tenant belongs.

"tenantName": "*Tenant name*
: Specifies the name of the tenant that comprises the encryption keys.

"passwordFile": "*Password file name*"
Specifies the password file that contains a password for accessing the RKM server. If you do not provide a password, then you are prompted for a password when the request is sent. A password must be 1 - 20 characters in length. For more information, see the **mmkeyserv** command in the IBM Spectrum Scale documentation.

Response data

```
{
    "jobs": [
        {
            "jobId": Job ID,
            "status": "RUNNING | COMPLETED | FAILED",
            "submitted": "Time and date",
            "completed": "Time and date",
            "runtime": Time,
            "request": {
                "type": "GET | POST | PUT | DELETE",
                "url": "Request URL"
            },
            "result": {
                "progress": [],
                "commands": [
                    "Command name "
                ],
                "stdout": [
                ],
                "stderr": [],
                "exitCode": Code            },
            "pids": []
        }
    ],
    "status": {
        "code": Status code,
        "message": "Status message"
    }
}
    }
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"jobId": "*Job ID*"
The unique ID of the job.

"status": "*RUNNING | COMPLETED | FAILED*"
The status of the job.

"submitted": "*Time and date*"
The time and date when the job was submitted.

"completed": "*Time and date*"
The time at which the job was completed.

"runtime": "*Time and date*"
The time that the job took to run.

"request"

"type": "*GET | POST | PUT | DELETE*"
The request type.

"URL": "*Request URL*"
The URL through which the job is submitted.

"result"

"progress": *Job progress*
Progress information for the request.

"commands": "*Command name*
Array of commands that are run in this job.

"stdout": "*message*
Request Information.

"info":"*Key details*"
The encryption key details.

"exitCode":"*Exit code*"
Exit code of command. Zero indicates success and any other value denotes failure.

"stderr":"*Error*"
CLI messages from stderr.

"pids": "*Process IDs*
The process IDs for the job.

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example gets information on the encryption key.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'  -d '{ \
   "serverName": "lodestar1.fyre.ibm.com",
  "tenantName": "devG1",
  "passwordFile": "/var/lib/mmfs/gui/passfile" \
 }' 'https://198.51.100.1:443/scalemgmt/v2/encryption/rkmKeys'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
    "jobs": [
        {
            "jobId": 1000000000001,
            "status": "COMPLETED",
            "submitted": "2021-06-18 16:50:17,335",
            "completed": "2021-06-18 16:50:19,999",
            "runtime": 2664,
            "request": {
                "type": "GET",
                "url": "/scalemgmt/v2/encryption/rkmKeys"
            },
            "result": {
                "progress": [],
                "commands": [
                    "mmkeyserv key show --server 'lodestar1.fyre.ibm.com' --server-pwd '/root/
passfile1' --tenant 'devG1' "
                ],
                "stdout": [
                    "KEY-fa079f5-9e276a98-4d51-4df0-aeff-f296a3cbd7d2",
                    "info: KEY-fa079f5-9e276a98-4d51-4df0-aeff-f296a3cbd7d2\n"
                ],
                "stderr": [],
                "exitCode": 0
            },
            "pids": []
        }
    ],
    "status": {
        "code": 200,
        "message": "The request finished successfully."
```

```
    }
}
```

Related reference

“mmkeyserv command” on page 469
Manages encryption key servers and clients.

Encryption/rkmServer: GET

Displays information about RKM servers.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET scalemgmt/v2/encryption/rkmServer` request displays information about remote key manager (RKM) servers. For more information about the fields in the data structures that are returned, see the "mmkeyserv command" on page 469.

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/encryption/rkmServer
```

where

rkmServer
Specifies the resource of this GET call.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Parameters

Table 55. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
keyServer	The hostname or IP address of the RKM server for which the details are displayed.	Required

Request data

No request data.

Response data

```
{
    "jobs": [
        {
            "jobId": Job ID,
            "status": "RUNNING | COMPLETED | FAILED",
            "submitted": "Time and date",
            "completed": "Time and date",
            "runtime": Time,
            "request": {
                "type": "GET | POST | PUT | DELETE",
                "url": "Request URL"
            },
            "result": {
                "progress": [],
                "commands": [
                    "Command name "
                ],
```

```
                "stdout": [
                 "type": "Server type"
                 "backupKeyServers": "Server type"
                 "distribute": "Keystore file"
                 "fips1402mode": "Password"
                 "interval": "Certification file label"
                 "ipa": "Tenant name"
                 "keyServer": "Tenant name"
                 "kmipCertificateExpiration": "Tenant name"
                 "label": "Tenant name"
                 "nistCompliance": "Tenant name"
                 "restCertificateExpiration": "Tenant name"
                 "restPort": "Port number"
                 "retry": "Number of attempts"
                 "timeout": "Number of attempts"
                 "userID": "Number of attempts"
                 ],
                "stderr": [],
                "exitCode": Code            },
            "pids": []
        }
    ],
    "status": {
        "code": Status code,
        "message": "Status message"
        }
}
    }
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"jobId": "*Job ID*"
The unique ID of the job.

"status": "*RUNNING | COMPLETED | FAILED*"
The status of the job.

"submitted": "*Time and date*"
The time and date when the job was submitted.

"completed": "*Time and date*"
The time at which the job was completed.

"runtime": "*Time and date*"
The time that the job took to run.

"request"

"type": "*GET | POST | PUT | DELETE*"
The request type.

"URL": "*Request URL*"
The URL through which the job is submitted.

"result"

"progress": *Job progress*
Progress information for the request.

"commands": "*Command name*
Array of commands that are run in this job.

"stdout": "*message*
Request Information.

"backupKeyServers": "*Back up servers*
The comma-separated list of server names that is added to the list of backup RKM servers in the RKM.conf file.

"distribute": "*yes | no*
Specifies whether the list of RKM server names (main RKM server and backup RKM servers) in the RKM.conf file are arranged in a different order on each node so that each node connects with the servers in a different order.

"fips1402mode": "*on | off*
Specifies whether the FIPS 140-2 certified encryption model is enabled for communications between the nodes.

"interval": "*Time*
The number of microseconds to wait between connection retries. The valid range is 1 - 1000000000. The default value is 10000 (0.1 seconds).

"ipa": "*IP address*
The server IP address.

"keyServer": "*Server name*
The RKM server name.

"kmipCertificateExpiration": "*Date and time*
The expiration date and time of the non-self-signed certificate files in a certificate chain that is used by the specified key server to establish communication on the KMIP port .

"label": "*Server label*
The label to identify the RKM server.

"nistCompliance": "*off |SP800-131A*
Specifies whether GPFS operates in the NIST 800-131A mode.

"restCertificateExpiration": "*Time and date*
The expiration date and time of the non-self-signed certificate files in a certificate chain that is used by the specified key server to establish communication on the REST port.

"restPort": "*REST Port number*
The port number for the Representational State Transfer (REST) interface.

"retry": "*Number of attempts*
The number of attempts to retry a connection to an RKM server. The valid range is 1 - 10 retries. The default value is three retries.

"timeout": "*Time*
The connection timeout, in seconds, for retrieving an MEK from an RKM server. The valid range is 1 - 120 seconds. The default value is 60 seconds.

"type": "*Server type*
The type of server.

"userID": "*REST user ID*"
The user ID for the RKM server. The default value is `SKLMAdmin`.

"exitCode":"*Exit code*"
Exit code of command. Zero indicates success and any other value denotes failure.

"stderr":"*Error*"
CLI messages from stderr.

"pids": "*Process IDs*
The process IDs for the job.

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example gets information on the RKM server.

Request data:

```
curl -X GET --header 'Content-Type: application/json' --header 'Accept: text/html' 'https://
198.51.100.1:443/scalemgmt/v2/encryption/rkmServer?keyServer=lodestar1.fyre.ibm.com'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.


```
{
    "jobs": [
        {
            "jobId": 3000000000011,
            "status": "COMPLETED",
            "submitted": "2021-06-21 05:22:28,005",
            "completed": "2021-06-21 05:22:28,309",
            "runtime": 304,
            "request": {
                "type": "GET",
                "url": "/scalemgmt/v2/encryption/rkmServer"
            },
            "result": {
                "progress": [],
                "commands": [
                    "mmkeyserv rkm show "
                ],
                "stdout": [
                {
                "serverList": [
                {
                "backupKeyServers": "",
                "distribute": "yes",
                "fips1402mode": "off",
                "interval": 10000,
                "ipa": "9.30.252.171",
                "keyServer": "lodestar1.fyre.ibm.com",
                "kmipCertificateExpiration": "2024-05-23 11:53:14 (+0000)",
                "label": "1_lodestar1",
                "nistCompliance": "on",
                "restCertificateExpiration": "2022-05-23 11:01:35 (+0000)",
                "restPort": "9443",
                "retry": 3,
                "timeout": 60,
                "type": "ISKLM",
               "userID": "SKLMAdmin"
               }
               ],
               "stderr": [],
                "exitCode": 0
            },
            "pids": []
        }
    ],
    "status": {
        "code": 200,
        "message": "The request finished successfully."
    }
}
```


Related reference

“mmkeyserv command” on page 469
Manages encryption key servers and clients.

Encryption/rkms: GET

Provides details about an RKM stanza.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET scalemgmt/v2/encryption/rkms` request displays information on a remote key manager (RKM) server stanza. The RKM stanza refers to the block of configuration information that describes a registered key client. It is stored in the `RKM.conf` file. For more information about the fields in the data structures that are returned, see the “mmkeyserv command” on page 469.

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/encryption/rkms
```

where

rkms
: Specifies the resource of this GET call.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request data

No request data.

Response data

```
{
    "jobs": [
        {
            "jobId": Job ID,
            "status": "RUNNING | COMPLETED | FAILED",
            "submitted": "Time and date",
            "completed": "Time and date",
            "runtime": Time,
            "request": {
                "type": "GET | POST | PUT | DELETE",
                "url": "Request URL"
            },
            "result": {
                "progress": [],
                "commands": [
                    "Command name "
                ],
                "stdout": [
                 "type": "Server type"
                 "kmipServerUri": "Server type"
                 "keyStore": "Keystore file"
                 "passphrase": "Password"
                 "clientCertLabel": "Certification file label"
                 "tenantName": "Tenant name
                ],
                "stderr": [],
                "exitCode": Code            },
            "pids": []
        }
    ],
    "status": {
```

```
        "code": Status code,
        "message": "Status message"
    }
}
    }
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"jobId": "*Job ID*"
The unique ID of the job.

"status": "*RUNNING | COMPLETED | FAILED*"
The status of the job.

"submitted": "*Time and date*"
The time and date when the job was submitted.

"completed": "*Time and date*"
The time at which the job was completed.

"runtime": "*Time and date*"
The time that the job took to run.

"request"

"type": "*GET | POST | PUT | DELETE*"
The request type.

"URL": "*Request URL*"
The URL through which the job is submitted.

"result"

"progress": *Job progress*
Progress information for the request.

"commands": "*Command name*
Array of commands that are run in this job.

"stdout": "*message*
Request Information.

"type": "*Server type*
The type of server.

"kmipServerUri": "*Server location*
The Server URL.

"keyStore": "*Keystore file*"
The keystore file.

"passphrase"
The Keystore password.

"clientCertLabel": "*Certification file label.*
Client certification file name.

"tenantName": "*Tenant name*"
The name of the tenant.

"exitCode":"*Exit code*"
Exit code of command. Zero indicates success and any other value denotes failure.

"stderr":"*Error*"
CLI messages from stderr.

"pids": "*Process IDs*
The process IDs for the job.

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example gets information on the RKM server stanza.

Request data:

```
curl -X GET --header 'Accept: application/json' --header 'Authorization: Basic
YWRtaW46YWRtaW4wMDE=' 'https://198.51.100.1:443/scalemgmt/v2/encryption/rkms'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
    "jobs": [
        {
            "jobId": 3000000000011,
            "status": "COMPLETED",
            "submitted": "2021-06-21 05:22:28,005",
            "completed": "2021-06-21 05:22:28,309",
            "runtime": 304,
            "request": {
                "type": "GET",
                "url": "/scalemgmt/v2/encryption/rkms"
            },
            "result": {
                "progress": [],
                "commands": [
                    "mmkeyserv rkm show "
                ],
                "stdout": [
                    "lodestar1_devG1 {",
                    "  type = ISKLM",
                    "  kmipServerUri = tls://9.30.252.171:5696",
                    "  keyStore = /var/mmfs/ssl/keyServ/serverKmip.1_lodestar1.myclient2.1.p12",
                    "  passphrase = Ar1cent@123456Nigam",
                    "  clientCertLabel = myclient2",
                    "  tenantName = devG1",
                    "}",
                    "info: lodestar1_devG1 {\n  type = ISKLM\n  kmipServerUri = tls://
9.30.252.171:5696\n  keyStore = /var/mmfs/ssl/keyServ/serverKmip.1_lodestar1.myclient2.1.p12\n
passphrase = Ar1cent@123456Nigam\n  clientCertLabel = myclient2\n  tenantName = devG1\n}\n"
                ],
                "stderr": [],
                "exitCode": 0
            },
            "pids": []
        }
    ],
    "status": {
        "code": 200,
        "message": "The request finished successfully."
    }
}
```

Related reference

“mmkeyserv command” on page 469
Manages encryption key servers and clients.

Encryption/rkmTenants: GET

Displays information about tenants and RKM servers.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET scalemgmt/v2/encryption/rkmTenants` request displays information about tenants and remote key manager (RKM) servers that they are associated with.. For more information about the fields in the data structures that are returned, see the "mmkeyserv command" on page 469.

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/encryption/rkmTenants
```

where

`rkmTenants`
Specifies the resource of this GET call.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Parameters

Table 56. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
keyServer	The name of the server where the tenant exists.	Optional
tenantName	The name of the tenant for which details are requested.	Optional

Request data

No request data.

Response data

```
{
    "jobs": [
        {
            "jobId": Job ID,
            "status": "RUNNING | COMPLETED | FAILED",
            "submitted": "Time and date",
            "completed": "Time and date",
            "runtime": Time,
            "request": {
                "type": "GET | POST | PUT | DELETE",
                "url": "Request URL"
            },
            "result": {
                "progress": [],
                "commands": [
```

```
                    "Command name "
                ],
                "stdout": [
                "keyServer": Key server name"
                "registeredClient": Client details
                "rkmId": "Server ID"
                "tenantName" : "Tenant name"
                ],
                "stderr": [],
                "exitCode": Code            },
            "pids": []
        }
    ],
    "status": {
        "code": Status code,
        "message": "Status message"
    }
}
    }
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"jobId": "*Job ID*"
The unique ID of the job.

"status": "*RUNNING | COMPLETED | FAILED*"
The status of the job.

"submitted": "*Time and date*"
The time and date when the job was submitted.

"completed": "*Time and date*"
The time at which the job was completed.

"runtime": "*Time and date*"
The time that the job took to run.

"request"

"type": "*GET | POST | PUT | DELETE*"
The request type.

"URL": "*Request URL*"
The URL through which the job is submitted.

"result"

"progress": *Job progress*
Progress information for the request.

"commands": "*Command name*
Array of commands that are run in this job.

"stdout": "*message*
Request Information.

"keyServer": *Server name*"
The server details.

"registeredClient": *Client details*
The client registered with the server.

"rkmId": "*Server ID*"
The ID of the RKM server.

"tenantName": "*Tenant name*"
The name of the tenant.

"exitCode":"*Exit code*"
Exit code of command. Zero indicates success and any other value denotes failure.

"stderr":"*Error*"
CLI messages from stderr.

"pids": "*Process IDs*
The process IDs for the job.

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example gets information on the tenant *devg1*.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json' 'https://198.51.100.1:443//
scalemgmt/v2/encryption/rkmTenant?keyServer=lodestar1.fyre.ibm.com&tenantName=devG1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
    "jobs": [
        {
            "jobId": 1000000000004,
            "status": "COMPLETED",
            "submitted": "2021-06-18 14:36:37,173",
            "completed": "2021-06-18 14:36:37,377",
            "runtime": 204,
            "request": {
                "type": "GET",
                "url": "/scalemgmt/v2/encryption/rkmTenants"
            },
            "result": {
                "progress": [],
                "commands": [
                    "mmkeyserv tenant show  --server 'lodestar1.fyre.ibm.com' "
                ],
                "stdout": [
                    "devG1",
                    "\tKey Server:          lodestar1.fyre.ibm.com",
                    "\tRegistered Client:   (none)",
                    "\tRKM Id:              (none)",
                    "info: devG1\n\tKey Server:lodestar1.fyre.ibm.com\n\tRegistered
                    Client:(none)\n\tRKM Id: (none)\n\n"
                ],
                "stderr": [],
                "exitCode": 0
            },
            "pids": []
        }
    ],
    "status": {
        "code": 200,
        "message": "The request finished successfully."
    }
}
```

Related reference

“mmkeyserv command” on page 469
Manages encryption key servers and clients.

Encryption/deregisterClient: PUT

Unregisters a key client from a tenant.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `/scalemgmt/v2/encryption/deregisterClient` request unregisters the key client from a tenant. For more information about the fields in the data structures that are returned, see the "mmkeyserv command" on page 469.

Request URL

```
https://<IP address or host name of API server>:port/scalemgmt/v2/encryption/deregisterClient
```

where:

deregisterClient
Specifies the resource to be created.

Request headers

```
Accept: application/json
```

Request data

```
{
  "clientName": "Key client name",
  "tenantName": "Tenant name",
  "passwordFile": "Password file name",

}
```

The details of the parameters are given in the following list:

"clientName": "*Key client name*"
Specifies the key client that must be unregistered.

"tenantName": "*Tenant name*"
Specifies the name of the tenant to which the key client belongs.

"passwordFile": "*Password file*"
Specifies the password file that contains a password for accessing the RKM server. If you do not provide a password, you are prompted for a password when the request is sent. A password must be 1 - 20 characters in length. For more information, see the **mmkeyserv** command in the IBM Spectrum Scale documentation.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"Commands issued",
             "progress":"Request progress,
```

```
            "exitCode":"Exit code",
            "stderr":"Error",
            "stdout":"CLI messages",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted": "Date and Time",
        "completed": "Date and Time",
        "runtime": "Duration",
        "status":"Job status",
        "pids":"Process IDs
        }
  ],
}
```

The details of the parameters are provided in the following list:

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*Commands issued*"
An array of commands that are run in this job.

"progress":"*Request progress*"
Specifies the progress information for the request.

"exitCode":"*Exit code*"
Specifies the exit code of command. Zero indicates success and any value other than zero denotes failure.

"stderr":"*Error*"
Specifies the CLI messages from stderr.

"stdout":"*String*"
Specifies the CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
Specifies the HTTP request type.

"url":"*URL*"
Specifies the URL through which the job is submitted.

"data":
Specifies the request data.

"jobId":"*ID*",
Specifies the unique ID of the job.

"submitted":"*Date and Time*"
Specifies the date and time at which the job was submitted.

"completed": "*Date and Time*"
Specifies the date and time at which the job was completed.

"runtime": "*Duration*"
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"pids":"Process ID"
Specifies the process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to unregister `myclient1` from a tenant.

Request data:

```
 curl -X PUT --header 'Content-Type: application/json' --header 'Accept: application/json' --
header 'Authorization: Basic YWRtaW46YWRtaW4wMDE=' -d '{ \
   "clientName": "myclient1"
   "tenantName": "devG1",
   "passwordFile": "/tmp/password",
     }' 'https://198.51.100.1:443/scalemgmt/v2/encryption/deregisterClient'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
 {
    "jobs": [
        {
            "jobId": 1000000000008,
            "status": "COMPLETED",
            "submitted": "2021-06-20 09:33:40,120",
            "completed": "2021-06-20 09:33:45,648",
            "runtime": 5528,
            "request": {
                "type": "POST",
                "url": "/scalemgmt/v2/encryption/deregisterClient"
            },
            "result": {
                "progress": [],
                "commands": [
                    "mmkeyserv client deregister 'myclient1' --tenant 'devG1' --server-pwd '/
root/passfile1' "
                ],
                "stdout": [
                    "mmkeyserv: Deleting the following KMIP certificate with label:
11856757298151928559_devG1_1624181101",
                    "mmkeyserv: Propagating the cluster configuration data to all",
                    "  affected nodes.  This is an asynchronous process.",
                    "info: mmkeyserv: Deleting the following KMIP certificate with label:
11856757298151928559_devG1_1624181101\n"
                ],
                "stderr": [],
                "exitCode": 0
            },
            "pids": []
        }
    ],
    "status": {
        "code": 200,
        "message": "The request finished successfully."
    }
}
```

Related reference

“mmkeyserv command” on page 469
Manages encryption key servers and clients.

Encryption/updateClientkeys: PUT

Replaces a valid or an expired client certificate in a specified key client.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `scalemgmt/v2/encryption/updateClientkeys` request replaces a client certificate, which has either expired or is still valid, for a specified key client. For more information about the fields in the data structures that are returned, see the “mmkeyserv command” on page 469.

Request URL

```
https://<IP address or host name of API server>:<port>scalemgmt/v2/encryption/updateClientkeys
```

where

updateClientkeys
Specifies the target of the PUT request.

Request headers

```
Accept: application/json
```

Request data

The following list of attributes is available in the request data:

```
{
  "clientName": "Client name",
  "newClientname": "New client name",
  "passwordFile": "Password file name",
  "daysToExpiration": Days till expiry,
  "keyStorePwdFile": "Keystore password file name"
  "clientCertFile": "Client certification file",
  "clientPrivateKeyFile": "Client private key file",
  "caCertFilePrefix": "Path and file name of certifcate prefix",
  "caCertChainFile": "CA certificates file
  "forceFlag":  true | false
  }
```

The details of the parameters are given in the following list.

"clientName": "*Client name*"
Specifies the name of the key client where you want to update the client certificate.

"newClientName": "*New client name*"
Specifies the new name of the key client. The name must be within 1 - 16 characters in length. It must be unique within the IBM Spectrum Scale cluster. If you do not provide a value for this parameter, then the certificate is not replaced with the new name.

"serverName": "*Server name*"
Specifies the name of the RKM server to which the key client belongs.

"passwordFile": "*Password file*"
The password file that comprises a password for accessing the RKM server.

"daysToExpiration": "*Number of days till expiration*"
The number of days until the new client certificate expires. The valid range is 1 - 18262. The default value is 1095.

"keyStorePwdFile" : "*Keystore password file*"
The password file that contains a client keystore password.

"clientCertFile": "*Client certificate file*"
The file that contains a client certificate from a certificate authority (CA).

"clientPrivateKeyFile" : "*Client private key file*"
The file that contains a client private key that matches the client certificate.

"caCertFilePrefix" : "*Path and file name of Certificate prefix*"
The path and file name prefix of non-self-signed certificate files in a certificate chain.

"caCertChainFile": "*CA Certificate file*"
The file that contains the certificates of the CA that signed the client certificate.

"forceflag": "*true | false*"
Specifies whether a self-signed client certificate is generated for the key client.

Response data

```
{
  "jobs": [
    {
      "jobId": ID of the job,
      "status": "Job status",
      "submitted": "Date and time when job was submitted",
      "completed": "Date and time when job was completed",
      "runtime": Time when Job ran,
      "request": {
        "type": "Request Type",
        "url": "Resource URL"
      },
      "result": {},
      "pids": []
    }
  ],
  "status": {
    "code": return status code,
    "message": "Return message."
  }
}
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"jobId": "*ID*",
The unique ID of the job.

"submitted": "*Time*"
The time at which the job was submitted.

"completed": "*Time*"
The time at which the job was completed.

"runtime": "*Time*"
The duration for which the job ran.

"status": "RUNNING | COMPLETED | FAILED"
Status of the job.

"result"
Array of commands that are run in this job.

"pids": *list*
A list of process IDs for this job.

"request"

"type": "{GET | POST | PUT | DELETE}"
HTTP request type.

"url": "*URL*"
The URL through which the job is submitted.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example replaces the client certificate.

Request data:

```
curl -X PUT --header 'Content-Type: application/json' --header 'Accept: application/json' --
header 'Authorization: Basic YWRtaW46VHJhY2VAMjAyMQ==' -d '{ \
    "clientName": "myclient1", \
   "newClientname": "myclient1", \
   "passwordFile": "/tmp/password", \
   "daysToExpiration": 1095, \
   "keyStorePwdFile": "/tmp/password", \
   "clientCertFile": "/tmp/cert", \
   "clientPrivateKeyFile": "/tmp/CA/certfiles.1.cert", \
   "caCertFilePrefix": "/tmp/cert", \
   "caCertChainFile": "/tmp/CA/certfiles.0.cert", \
   "forceFlag": false \
   }' 'https://198.51.100.1:443/scalemgmt/v2/encryption/updateClientkeys'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
    "jobs": [
        {
            "jobId": 3000000000012,
            "status": "COMPLETED",
            "submitted": "2021-06-21 05:30:06,455",
            "completed": "2021-06-21 05:30:23,457",
            "runtime": 17002,
            "request": {
                "data": {
                    "clientName": "myclient2",
                    "daysToExpiration": 1095,
                    "keyStorePwdFile": "/home/passfile1",
                    "newClientname": "myclient1",
                    "passwordFile": "/home/passfile1"
                    "clientCertFile": "/tmp/cert", \
                    "clientPrivateKeyFile": "/tmp/CA/certfiles.1.cert", \
                    "caCertFilePrefix": "/tmp/cert", \
                    "caCertChainFile": "/tmp/CA/certfiles.0.cert", \
                    "forceFlag": false
                },
                "type": "PUT",
                "url": "/scalemgmt/v2/encryption/updateClientkeys"
            },
            "result": {
                "progress": [],
                "commands": [
                    "mmkeyserv client update 'myclient2' --client 'myclient1' --days 1095 --
keystore-pwd '/home/passfile1' --server-pwd '/home/passfile1' "
                ],
                "stdout": [
                    "mmkeyserv: [I] Client currently does not have access to the key.  Continue
the registration process ...",
                    "mmkeyserv: Successfully accepted client certificate",
                    "mmkeyserv: Propagating the cluster configuration data to all",
                    "  affected nodes.  This is an asynchronous process.",
                    "mmkeyserv: Deleting the following KMIP certificate with label:
9842179411678971055_devG1_1624267300",
```

```
                    "info: mmkeyserv: [I] Client currently does not have access
to the key.  Continue the registration process ...\nmmkeyserv: Successfully accepted
client certificate\nmmkeyserv: Deleting the following KMIP certificate with label:
9842179411678971055_devG1_1624267300\n"
                ],
                "stderr": [],
                "exitCode": 0
            },
            "pids": []
        }
    ],
    "status": {
        "code": 200,
        "message": "The request finished successfully."
    }
}
```

Related reference

“mmkeyserv command” on page 469
Manages encryption key servers and clients.

Encryption/serverUpdate: PUT

Updates an RKM server connection to the IBM Spectrum Scale Erasure Code Edition cluster.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `scalemgmt/v2/encryption/serverUpdate` request updates a remote key manager (RKM) server connection to the IBM Spectrum Scale cluster. . For more information about the fields in the data structures that are returned, see the "mmkeyserv command" on page 469.

Request URL

```
https://<IP address or host name of API server>:<port>scalemgmt/v2/encryption/serverUpdate
```

where

`serverUpdate`
Specifies the resource that must be updated.

Request headers

```
Accept: application/json
```

Parameters

Table 57. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale Erasure Code Edition system to perform the requested operation	Required

Request data

The following list of attributes is available in the request data:

```
{
 "restPortNumber": Port number,
  "restUserID": "User ID ",
  "passwordFile": "Password file name",
  "accept": true | false,
  "certFilePrefix": "Certificate File prefix",
  "serverNamebackup": "Server name list",
  "dis": true | false,
  "nodis": true | false,
  "connectionTimeout": Time,
  "connectionAttempts": Number of attempts,
  "microseconds": Time
  }
```

The details of the parameters are given in the following list.

"serverName": "*Server name*"
Specifies the hostname or the IP address of the RKM server.

"restPortNumber": "*REST Port Number*"
Specifies the port number for the Representational State Transfer (REST) interface on the IBM Security Key Lifecycle Manager server.

"restUserID": "*User ID*"
Specifies the user ID for the RKM server. The default value is SKLMAdmin.

"passwordFile": "*Password file*"
Specifies the password file that contains a password for accessing the RKM server. If you do not provide a password, then you are prompted for a password when the request is sent. A password must be 1 - 20 characters in length. For more information, see the **mmkeyserv** command in the IBM Spectrum Scale documentation.

"accept": "*true | false*"
Specifies whether the command is configured to automatically accept certificates from the RKM server.

"caCertFilePrefix" : "*Path and file name of the Certificate file prefix* "
The path and file name prefix of non-self-signed certificate files in a certificate chain.

"serverNamebackup" : "*Server name list*"
Specifies a comma-separated list of server names that you want to add to the list of backup RKM servers defined in the RKM.conf file.

"dis": "*true | false*"
Specifies whether the list of RKM server names, including the main RKM server and backup RKM servers, are arranged in the RKM.conf file in a different order on each node. This arrangement ensures that each node connects with the servers in a different order.

"nodis": "*true | false*"
Specifies whether the list of RKM server names is arranged in the RKM.conf file.

"connectionTimeout": "*Time*"
Specifies the connection timeout, in seconds, for retrieving a master encryption key (MEK) from an RKM server. The valid range is 1 - 120 seconds and the default value is 60 seconds.

"connectionAttempts": "*Number of attempts*"
Specifies the number of attempts to retry a connection to an RKM server. The valid range is 1 - 10 retries and the default value is three retries.

"microseconds": "*Time*"
Specifies the number of microseconds of waiting between attempts to connect. The valid range is 1 - 1000000000 and the default value is 10000 (0.1 seconds).

Response data

```
{
  "jobs": [
    {
      "jobId": ID of the job,
      "status": "Job status",
      "submitted": "Date and time when job was submitted",
      "completed": "Date and time when job was completed",
      "runtime": Time when Job ran,
      "request": {
        "type": "Request Type",
        "url": "Resource URL"
      },
      "result": {},
         "progress": [],
        "commands": [
          ""
        ],
        "stdout": [
          " "
        ],
        "stderr": [],
        "exitCode": 0
```

```
      },
        "pids": []
    }
  ],
  "status": {
    "code": return status code,
    "message": "Return message."
  }
}
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"jobId": "*ID*",
The unique ID of the job.

"submitted": "*Time*"
The time at which the job was submitted.

"completed": "*Time*"
The time at which the job was completed.

"runtime": "*Time*"
The duration for which the job ran.

"status": "RUNNING | COMPLETED | FAILED"
Status of the job.

"result"

"progress": *Job progress*
Progress information for the request.

"commands": "*Command name*
Array of commands that are run in this job.

"stdout": "*message*
Request Information.

"exitCode":"*Exit code*"
Exit code of command. Zero indicates success and any other value denotes failure.

"stderr":"*Error*"
CLI messages from stderr.

"pids": *list*
A list of process IDs for this job.

"request"

"type": "{GET | POST | PUT | DELETE}"
HTTP request type.

"url": "*URL*"
The URL through which the job is submitted.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example updates the connection of the *lodestar1.fyre.ibm.com* server with the IBM Spectrum Scale cluster.

Request data:

```
curl -X PUT --header 'Content-Type: application/json' --header 'Accept: application/json' --
header 'Authorization: Basic YWRtaW46VHJhY2VAMjAyMQ==' -d '{ \
  "serverName": "lodestar1.fyre.ibm.com",
  "restPortNumber": 44742,
  "restUserID": "admin",
  "passwordFile": "/var/lib/mmfs/gui/passfile",
  "accept": false,
  "certFilePrefix": "CertFilesPrefix.n.cert",
  "serverNamebackup": "string",
  "dis": true,
  "nodis": false,
  "connectionTimeout": 60,
  "connectionAttempts": 3,
  "microseconds": 10000
   }\
 }' 'https://198.51.100.1:443/scalemgmt/v2/encryption/serverUpdate'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs": [
    {
      "jobId": 1000000000003,
      "status": "COMPLETED",
      "submitted": "2021-06-18 08:14:31,854",
      "completed": "2021-06-18 08:14:40,799",
      "runtime": 8945,
      "request": {
        "type": "PUT",
        "url": "/scalemgmt/v2/encryption/serverUpdate"
      },
      "result": {
        "progress": [],
        "commands": [
          "mmkeyserv server update 'lodestar1.fyre.ibm.com' --server-pwd '/root/passfile1' --
accept "
        ],
        "stdout": [
          "mmkeyserv: Propagating the cluster configuration data to all",
          "  affected nodes.  This is an asynchronous process.",
          "info: "
        ],
        "stderr": [],
        "exitCode": 0
      },
      "pids": []
    }
  ],
  "status": {
    "code": 200,
    "message": "The request finished successfully."
  }
}
```

Related reference

“mmkeyserv command” on page 469
Manages encryption key servers and clients.

Encryption/serverAdd: POST

Adds an RKM server connection to the IBM Spectrum Scale Erasure Code Edition cluster.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `scalemgmt/v2/encryption/serverAdd` request adds a remote key manager (RKM) server connection to the IBM Spectrum Scale cluster. For more information about the fields in the data structures that are returned, see the "mmkeyserv command" on page 469.

Request URL

```
https://<IP address or host name of API server>:<port>scalemgmt/v2/encryption/serverAdd
```

where

serverAdd
Specifies the resource that must be added.

Request headers

```
Accept: application/json
```

Parameters

Table 58. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale Erasure Code Edition system to perform the requested operation.	Required

Request data

The following list of attributes is available in the request data:

```
{
 "restPortNumber": Port number,
  "restUserID": "User ID ",
  "passwordFile": "Password file name",
  "accept": true | false,
  "certFilePrefix": "Certificate File prefix",
  "serverNamebackup": "Server name list",
  "dis": true | false,
  "nodis": true | false,
  "connectionTimeout": Time,
  "connectionAttempts": Number of attempts,
  "microseconds": Time
  }
```

The details of the parameters are given in the following list.

"serverName": "*Server name*"
Specifies the hostname or the IP address of the RKM server.

"restPortNumber": "*REST Port Number*"
Specifies the port number for the Representational State Transfer (REST) interface on the IBM Security Key Lifecycle Manager server.

"restUserID": "*User ID*"
Specifies the user ID for the RKM server. The default value is SKLMAdmin.

"passwordFile": "*Password file*"
Specifies the password file that contains a password for accessing the RKM server. If you do not provide a password, then you are prompted for a password when the request is sent. A password must be 1 - 20 characters in length. For more information, see the **mmkeyserv** command in the IBM Spectrum Scale documentation.

"accept": "*true | false*"
Specifies whether the command is configured to automatically accept certificates from the RKM server.

"caCertFilePrefix" : "*Path and file name of the Certificate file prefix* "
The path and file name prefix of non-self-signed certificate files in a certificate chain.

"serverNamebackup" : "*Server name list*"
Specifies a comma-separated list of server names that you want to add to the list of backup RKM servers defined in the RKM.conf file.

"dis": "*true | false*"
Specifies whether the list of RKM server names, including the main RKM server and backup RKM servers, are arranged in the RKM.conf file in a different order on each node. This arrangement ensures that each node connects with the servers in a different order.

"nodis": "*true | false*"
Specifies whether the list of RKM server names is arranged in the RKM.conf file.

"connectionTimeout": "*Time*"
Specifies the connection timeout, in seconds, for retrieving a master encryption key (MEK) from an RKM server. The valid range is 1 - 120 seconds and the default value is 60 seconds.

"connectionAttempts": "*Number of attempts*"
Specifies the number of attempts to retry a connection to an RKM server. The valid range is 1 - 10 retries and the default value is three retries.

"microseconds": "*Time*"
Specifies the number of microseconds of waiting between attempts to connect. The valid range is 1 - 1000000000 and the default value is 10000 (0.1 seconds).

Response data

```
{
  "jobs": [
    {
      "jobId": ID of the job,
      "status": "Job status",
      "submitted": "Date and time when job was submitted",
      "completed": "Date and time when job was completed",
      "runtime": Time when Job ran,
      "request": {
        "type": "Request Type",
        "url": "Resource URL"
      },
      "result": {},
         "progress": [],
        "commands": [
          ""
        ],
        "stdout": [
          " "
        ],
        "stderr": [],
        "exitCode": 0
```

```
      },
        "pids": []
    }
  ],
  "status": {
    "code": return status code,
    "message": "Return message."
  }
}
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"jobId": "*ID*",
The unique ID of the job.

"submitted": "*Time*"
The time at which the job was submitted.

"completed": "*Time*"
The time at which the job was completed.

"runtime": "*Time*"
The duration for which the job ran.

"status": "RUNNING | COMPLETED | FAILED"
Status of the job.

"result"

"progress": *Job progress*
Progress information for the request.

"commands": "*Command name*
Array of commands that are run in this job.

"stdout": "*message*
Request Information.

"exitCode":"*Exit code*"
Exit code of command. Zero indicates success and any other value denotes failure.

"stderr":"*Error*"
CLI messages from stderr.

"pids": *list*
A list of process IDs for this job.

"request"

"type": "{GET | POST | PUT | DELETE}"
HTTP request type.

"url": "*URL*"
The URL through which the job is submitted.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example adds the *lodestar1.fyre.ibm.com* server connection with IBM Spectrum Scale cluster.

Request data:

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --
header 'Authorization: Basic YWRtaW46VHJhY2VAMjAyMQ==' -d '{ \
  "serverName": "lodestar1.fyre.ibm.com",
  "restPortNumber": 44742,
  "restUserID": "admin",
  "passwordFile": "/var/lib/mmfs/gui/passfile",
  "accept": false,
  "certFilePrefix": "CertFilesPrefix.n.cert",
  "serverNamebackup": "string",
  "dis": true,
  "nodis": false,
  "connectionTimeout": 60,
  "connectionAttempts": 3,
  "microseconds": 10000
   }\
 }' 'https://198.51.100.1:443/scalemgmt/v2/encryption/serverAdd'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs": [
    {
      "jobId": 1000000000003,
      "status": "COMPLETED",
      "submitted": "2021-06-18 08:14:31,854",
      "completed": "2021-06-18 08:14:40,799",
      "runtime": 8945,
      "request": {
        "type": "POST",
        "url": "/scalemgmt/v2/encryption/serverAdd"
      },
      "result": {
        "progress": [],
        "commands": [
          "mmkeyserv server add 'lodestar1.fyre.ibm.com' --server-pwd '/root/passfile1' --
accept "
        ],
        "stdout": [
          "mmkeyserv: Propagating the cluster configuration data to all",
          "  affected nodes.  This is an asynchronous process.",
          "info: "
        ],
        "stderr": [],
        "exitCode": 0
      },
      "pids": []
    }
  ],
  "status": {
    "code": 200,
    "message": "The request finished successfully."
  }
}
```

Related reference

“mmkeyserv command” on page 469
Manages encryption key servers and clients.

Encryption/addTenant: POST

Adds a tenant on RKM servers.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `/scalemgmt/v2/encryption/addTenant` request adds a tenant on the remote key manager (RKM) server. A tenant is an IBM Security Key Lifecycle Manager device group that comprises the encryption keys for registered key clients. For more information about the fields in the data structures that are returned, see the “mmkeyserv command” on page 469.

Request URL

```
https://<IP address or host name of API server>:port/scalemgmt/v2/encryption/addTenant
```

where:

addTenant
Specifies the resource to be created.

Request headers

```
Accept: application/json
```

Request data

```
{

  "tenantName": "Tenant name",
  "serverName": "Server name"
  "passwordFile": "Password file name",

}
```

The details of the parameters are given in the following list:

"tenantName": "*Tenant name*"
Specifies the name of the tenant that you want to add to the RKM server.

"serverName": "*Server name*"
Specifies the name of the RKM server to which the tenant belongs.

"passwordFile": "*Password file*"
Specifies the password file that contains a password for accessing the RKM server. If you do not provide a password, you are prompted for a password when the request is sent. A password must be 1 - 20 characters in length. For more information, see the **mmkeyserv** command in the IBM Spectrum Scale documentation.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
```

```
          {
            "commands":"Commands issued",
            "progress":"Request progress,
            "exitCode":"Exit code",
            "stderr":"Error",
            "stdout":"CLI messages",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted": "Date and Time",
        "completed": "Date and Time",
        "runtime": "Duration",
        "status":"Job status",
        "pids":"Process IDs
        }
  ],
}
```

The details of the parameters are provided in the following list:

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*Commands issued*"
An array of commands that are run in this job.

"progress":"*Request progress*"
Specifies the progress information for the request.

"exitCode":"*Exit code*"
Specifies the exit code of command. Zero indicates success and any value other than zero denotes failure.

"stderr":"*Error*"
Specifies the CLI messages from stderr.

"stdout":"*String*"
Specifies the CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
Specifies the HTTP request type.

"url":"*URL*"
Specifies the URL through which the job is submitted.

"data":
Specifies the request data.

"jobId":"*ID*",
Specifies the unique ID of the job.

"submitted":"*Date and Time*"
Specifies the date and time at which the job was submitted.

"completed":*Date and Time*"
Specifies the date and time at which the job was completed.

"runtime":*Duration*"
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"pids":"Process ID"
Specifies the process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to add a tenant.

Request data:

```
 curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --
header 'Authorization: Basic YWRtaW46YWRtaW4wMDE=' -d '{ \
   "tenantName": "devG1",
   "serverName": "sklm11.fyre.ibm.com",
   "passwordFile": "/tmp/password",
   }' 'https://198.51.100.1:443/scalemgmt/v2/encryption/addTenant'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
    "jobs": [
        {
            "jobId": 1000000000002,
            "status": "COMPLETED",
            "submitted": "2021-06-18 14:25:03,238",
            "completed": "2021-06-18 14:25:13,280",
            "runtime": 10042,
            "request": {
                "data": {
                    "passwordFile": "/root/passfile1",
                    "serverName": "lodestar1.fyre.ibm.com",
                    "tenantName": "devG1"
                },
                "type": "POST",
                "url": "/scalemgmt/v2/encryption/addTenant"
            },
            "result": {
                "progress": [],
                "commands": [
                    "mmkeyserv tenant add 'devG1' --server 'lodestar1.fyre.ibm.com' --server-
pwd '/root/passfile1' "
                ],
                "stdout": [
                    "mmkeyserv: Propagating the cluster configuration data to all",
                    "  affected nodes.  This is an asynchronous process.",
                    "info: "
                ],
                "stderr": [],
                "exitCode": 0
            },
            "pids": []
        }
    ],
    "status": {
        "code": 200,
        "message": "The request finished successfully."
    }
}
```

Related reference

“mmkeyserv command” on page 469
Manages encryption key servers and clients.

Encryption/createKey: POST

Creates encryption keys within a tenant.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `/scalemgmt/v2/encryption/createKey` request creates an encryption key within a tenant. The key IDs are also displayed on the remote key manager (RKM) server console. For more information about the fields in the data structures that are returned, see the "mmkeyserv command" on page 469.

Request URL

```
https://<IP address or host name of API server>:port/scalemgmt/v2/encryption/createKey
```

where:

createKey
Specifies the resource to be created.

Request headers

```
Accept: application/json
```

Request data

```
{
  "tenantName": "Tenant name",
  "serverName": "Server name",
  "passwordFile": "Password file name",
  "numberOfKeys": Number of keys created
}
```

The details of the parameters are given in the following list.

"tenantName": "*Tenant name*"
Specifies the name of the tenant where the encryption key or keys are created.

"serverName": "*Server name*"
Specifies the hostname or the IP address of the RKM server.

"passwordFile": "*Password file*"
Specifies the password file that contains a password for accessing the RKM server. If you do not provide a password, then you are prompted for a password when the request is sent. A password must be 1 - 20 characters in length. For more information, see the **mmkeyserv** command in the IBM Spectrum Scale documentation.

"numberOfKeys": "*Number of keys created*"
Specifies the number of key or keys that are created. The default value is 1.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
```

```
    {
       "result":"",
         {
           "commands":"Commands issued",
           "progress":"Request progress,
           "exitCode":"Exit code",
           "stderr":"Error",
           "stdout":"CLI messages",
         },
       "request":" ",
         {
           "type":"{GET | POST | PUT | DELETE}",
           "url":"URL",
           "data":"",
         }
       "jobId":"ID",
       "submitted": "Date and Time",
       "completed": "Date and Time",
       "runtime": "Duration",
       "status":"Job status",
       "pids":"Process IDs
       }
 ],
}
```

The details of the parameters are provided in the following list:

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*Commands issued*"
An array of commands that are run in this job.

"progress":"*Request progress*"
Specifies the progress information for the request.

"exitCode":"*Exit code*"
Specifies the exit code of command. Zero indicates success and any value other than zero denotes failure.

"stderr":"*Error*"
Specifies the CLI messages from stderr.

"stdout":"*String*"
Specifies the CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
Specifies the HTTP request type.

"url":"*URL*"
Specifies the URL through which the job is submitted.

"data":
Specifies the request data.

"jobId":"*ID*",
Specifies the unique ID of the job.

"submitted":"*Date and Time*"
Specifies the date and time at which the job was submitted.

"completed":*Date and Time*"
Specifies the date and time at which the job was completed.

"runtime":*Duration*"
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"pids":"Process ID"
Specifies the process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to create an encryption key for the tenant devG1.

Request data:

```
 curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --
header 'Authorization: Basic YWRtaW46YWRtaW4wMDE=' -d '{ \
   "tenantName": "devG1",
  "serverName": "sklm11.fyre.ibm.com",
  "passwordFile": "/tmp/password",
  "numberOfKeys": 1 \
 }' 'https://198.51.100.1:443/scalemgmt/v2/encryption/createKey'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000004,
    "status" : "COMPLETED",
    "submitted" : "2021-01-30 18:01:45,173",
    "completed" : "2021-01-30 18:01:45,737",
    "runtime" : 564,
    "request" : {
      "type" : "POST",
      "url" : "scalemgmt/v2/encryption/createKey"
      "data": " "
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ mmkeyserv key create --server 'sklm11.fyre.ibm.com' [--server-pwd '/tmp/
password']
   --tenant 'devG1' [--count '1'] ],
      "stdout" : [ " " ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Related reference

“mmkeyserv command” on page 469
Manages encryption key servers and clients.

Encryption/clients: POST

Creates a key client to connect with the remote key management (RKM) server.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `/scalemgmt/v2/encryption/clients` request creates a key client that communicates with the RKM server. For more information about the fields in the data structures that are returned, see the "mmkeyserv command" on page 469.

Request URL

```
https://<IP address or host name of API server>:port/scalemgmt/v2/encryption/clients
```

where:

clients
Specifies the resource to be added.

Request headers

```
Accept: application/json
```

Request data

```
{
  "clientName": "Client name",
  "serverName": "Server name",
  "passwordFile": "Password file path",
  "daysToExpiration": Number of days till expiration,
  "keyStorePwdFile": "Keystore password file",
  "clientCertFile": "Client certification file",
  "clientPrivateKeyFile": "Client private key file",
  "caCertFilePrefix": "Path and file name of certifcate prefix",
  "caCertChainFile": "CA certificates file"
}
```

The details of the parameters are given in the following list.

"clientName": "*Client name*"
Specifies the name of the key client that is created. The name must be within 1 - 16 characters in length. It must be unique within the IBM Spectrum Scale cluster.

"serverName": "*Server name*"
Specifies the name of the RKM server to which the key client belongs.

"passwordFile": "*Password file*"
Specifies the password file that contains a password for accessing the RKM server. If you do not provide a password, then you are prompted for one when the request is sent. A password must be 1 - 20 characters in length. For more information, see the **mmkeyserv** command in the IBM Spectrum Scale documentation.

"daysToExpiration": "*Number of days till expiration*"
The number of days until the new client certificate expires. The valid range is 1 - 18262. The default value is 1095.

"keyStorePwdFile" : "*Keystore password file*"
The password file that contains a client keystore password.

"clientCertFile": "*Client certificate file*"
The file that contains a client certificate from a certificate authority (CA).

"clientPrivateKeyFile" : "*Client private key file*"
The file that contains a client private key that matches the client certificate.

"caCertFilePrefix" : "*Path and file name of Certificate prefix*"
The path and file name prefix of non-self-signed certificate files in a certificate chain.

"caCertChainFile": "*CA Certificate file*"
The file that contains the certificates of the CA that signed the client certificate.

Response data

```
{
  "jobs": [
    {
      "jobId": 3000000000003,
      "status": "RUNNING | COMPLETED | FAILED",
      "submitted": "Time and date",
      "completed": "Time and date",
      "runtime": 7,
      "request": {
        "data": {
          "caCertChainFile": "Password file name",
          "caCertFilePrefix": "Prefix file name",
          "clientCertFile": "Certificate file name",
          "clientName": "Client name",
          "clientPrivateKeyFile": "Private key file",
          "daysToExpiration": Number of days,
          "keyStorePwdFile": "Keystore Password file",
          "passwordFile": "Password file name",
          "serverName": "Server name"
        },
        "type": "POST | GET | PUT | DELETE",
        "url": "Request URL"
      },
      "result": {},
      "pids": []
    }
  ],
  "status": {
    "code": Request Code,
    "message": "Request message"
  }
}
```

The details of the parameters are provided in the following list:

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*Commands issued*"
An array of commands that are run in this job.

"progress":"*Request progress*"
Specifies the progress information for the request.

"exitCode":"*Exit code*"
Specifies the exit code of command. Zero indicates success and any value other than zero denotes failure.

"stderr":"*Error*"
Specifies the CLI messages from stderr.

"stdout":"*String*"
Specifies the CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
Specifies the HTTP request type.

"url":"*URL*"
Specifies the URL through which the job is submitted.

"data":
Specifies the request data.

"caCertChainFile": "*CA Certificate file*"
The file that contains the certificates of the CA that signed the client certificate.

"caCertFilePrefix" : "*Path and file name of Certificate prefix*"
The path and file name prefix of non-self-signed certificate files in a certificate chain.

"clientCertFile": "*Client certificate file*"
The file that contains a client certificate from a certificate authority (CA).

"clientName": "*Client name*"
Specifies the name of the key client that is created. The name must be within 1 - 16 characters in length. It must be unique within the IBM Spectrum Scale cluster. Required.

"clientPrivateKeyFile" : "*Client private key file*"
The file that contains a client private key that matches the client certificate.

"daysToExpiration": "*Number of days till expiration*"
The number of days until the new client certificate expires. The valid range is 1 - 18262. The default value is 1095.

"keyStorePwdFile": "*Keystore password file*"
The password file that contains a client keystore password.

"passwordFile": "*Password file*"
Specifies the password file that contains a password for accessing the RKM server. If you do not provide a password, then you are prompted for one when the request is sent. A password must be 1 - 20 characters in length. For more information, see the **mmkeyserv** command in the IBM Spectrum Scale documentation.

"passwordFile": "*Password file*"
Specifies the password file that contains a password for accessing the RKM server. If you do not provide a password, then you are prompted for one when the request is sent. A password must be 1 - 20 characters in length. For more information, see the **mmkeyserv** command in the IBM Spectrum Scale documentation.

"serverName": "*Server name*"
Specifies the name of the RKM server to which the key client belongs. Required.

"jobId":"*ID*",
Specifies the unique ID of the job.

"submitted":"*Date and Time*"
Specifies the date and time at which the job was submitted.

"completed":*Date and Time*"
Specifies the date and time at which the job was completed.

"runtime":*Duration*"
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"pids":"Process ID"
Specifies the process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to create a key client `myclient1`.

Request data:

```
 curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --
header 'Authorization: Basic YWRtaW46YWRtaW4wMDE=' -d '{ \
   "clientName": "myclient1", \
   "serverName": "sklm11.fyre.ibm.com", \
   "passwordFile": "/tmp/password", \
   "daysToExpiration": 1095, \
   "keyStorePwdFile": "/tmp/password", \
   "clientCertFile": "/tmp/cert", \
   "clientPrivateKeyFile": "/tmp/CA/certfiles.1.cert", \
   "caCertFilePrefix": "/tmp/cert", \
   "caCertChainFile": "/tmp/CA/certfiles.0.cert" \
 }' 'https://198.51.100.1:443/scalemgmt/v2/encryption/clients'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
    "jobs": [
        {
            "jobId": 1000000000002,
            "status": "COMPLETED",
            "submitted": "2021-06-20 06:45:11,894",
            "completed": "2021-06-20 06:45:20,141",
            "runtime": 8247,
            "request": {
                "data": {
                    "clientName": "myclient2",
                    "keyStorePwdFile": "/root/passfile1",
                    "passwordFile": "/root/passfile1",
                    "serverName": "lodestar1.fyre.ibm.com"
                    "clientCertFile": "/tmp/cert", \
                    "clientPrivateKeyFile": "/tmp/CA/certfiles.1.cert", \
                    "caCertFilePrefix": "/tmp/cert", \
                    "caCertChainFile": "/tmp/CA/certfiles.0.cert" \
                },
                "type": "POST",
                "url": "/scalemgmt/v2/encryption/clients"
            },
            "result": {
                "progress": [],
                "commands": [
                    "mmkeyserv client create 'myclient2' --server 'lodestar1.fyre.ibm.com' --
server-pwd '/root/passfile1' --keystore-pwd '/root/passfile1' "
                ],
                "stdout": [
                    "mmkeyserv: Propagating the cluster configuration data to all",
                    "  affected nodes.  This is an asynchronous process.",
                    "info: "
                ],
                "stderr": [],
                "exitCode": 0
            },
            "pids": []
        }
    ],
    "status": {
        "code": 200,
        "message": "The request finished successfully."
    }
}
```

Related reference

“mmkeyserv command” on page 469

Manages encryption key servers and clients.

Encryption/registerClient: POST

Registers a key client to a tenant.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `/scalemgmt/v2/encryption/registerClient` request registers a key client to a tenant. For more information about the fields in the data structures that are returned, see the "mmkeyserv command" on page 469.

Request URL

```
https://<IP address or host name of API server>:port/scalemgmt/v2/encryption/registerClient
```

where:

registerClient
Specifies the resource to be created. Required.

Request headers

```
Accept: application/json
```

Request data

```
{
  "clientName": "Key client name",
  "tenantName": "Tenant name",
  "passwordFile": "Password file name",
  "rkmID": "string"
}
```

The details of the parameters are given in the following list.

"clientName": "*Key client name*"
Specifies the key client that you want to register. The key client name must be within 1 - 16 characters in length. It must be unique within the IBM Spectrum Scale cluster. Required.

"tenantName": "*Tenant name*"
Specifies the name of the tenant to which the key client belongs.

"passwordFile": "*Password file*"
Specifies the file that contains a password for accessing the RKM server. If you do not provide a password, you are prompted for it when the request is sent. A password must be 1 - 20 characters in length. For more information, see the **mmkeyserv** command in the IBM Spectrum Scale documentation.

"rkmID":"*Server ID*"
Specifies a new remote key manager (RKM) ID. An RKM ID must be unique within the cluster. It must be 1 - 21 characters in length and contain only alphanumeric characters or an underscore (_). It must begin with a letter or an underscore. An RKM ID identifies an RKM stanza in the `RKM.conf` file. The stanza contains the information that a node needs to retrieve a master encryption key (MEK) from an RKM.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"Commands issued",
             "progress":"Request progress,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"CLI messages",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted": "Date and Time",
         "completed": "Date and Time",
         "runtime": "Duration",
         "status":"Job status",
         "pids":"Process IDs
         }
   ],
  }
```

The details of the parameters are provided in the following list:

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*Commands issued*"
An array of commands that are run in this job.

"progress":"*Request progress*"
Specifies the progress information for the request.

"exitCode":"*Exit code*"
Specifies the exit code of command. Zero indicates success and any value other than zero denotes failure.

"stderr":"*Error*"
Specifies the CLI messages from stderr.

"stdout":"*String*"
Specifies the CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
Specifies the HTTP request type.

"url":"*URL*"
Specifies the URL through which the job is submitted.

"data":
Specifies the request data.

"jobId":"*ID*",
Specifies the unique ID of the job.

"submitted":"*Date and Time*"
Specifies the date and time at which the job was submitted.

"completed":*Date and Time*"
Specifies the date and time at which the job was completed.

"runtime":*Duration*"
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"pids":"Process ID"
Specifies the process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to register `myclient2` to a tenant.

Request data:

```
 curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --
header 'Authorization: Basic YWRtaW46YWRtaW4wMDE=' -d '{ \
   "clientName": "myclient2"
   "tenantName": "devG1",
   "passwordFile": "/tmp/password",
    "rkmID": "AU8763hgsu09"
   }' 'https://198.51.100.1:443/scalemgmt/v2/encryption/registerClient'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
    "jobs": [
        {
            "jobId": 1000000000007,
            "status": "COMPLETED",
            "submitted": "2021-06-20 09:25:00,801",
            "completed": "2021-06-20 09:25:12,142",
            "runtime": 11341,
            "request": {
                "data": {
                    "clientName": "myclient1",
                    "passwordFile": "/root/passfile1",
                    "rkmID": "lodestar1_devG1",
                    "tenantName": "devG1"
                },
                "type": "POST",
                "url": "/scalemgmt/v2/encryption/registerClient"
            },
            "result": {
                "progress": [],
                "commands": [
                    "mmkeyserv client register 'myclient1' --rkm-id 'lodestar1_devG1' --tenant
'devG1' --server-pwd '/root/passfile1' "
                ],
                "stdout": [
                    "mmkeyserv: [I] Client currently does not have access to the key.  Continue
the registration process ...",
                    "mmkeyserv: Successfully accepted client certificate",
                    "mmkeyserv: Propagating the cluster configuration data to all",
                    "  affected nodes.  This is an asynchronous process.",
                    "info: mmkeyserv: [I] Client currently does not have access to the key.
Continue the registration process ...\nmmkeyserv: Successfully accepted client certificate\n"
                ],
                "stderr": [],
                "exitCode": 0
            },
            "pids": []
```

```
        }
    ],
    "status": {
        "code": 200,
        "message": "The request finished successfully."
    }


}
```

Related reference

“mmkeyserv command” on page 469
Manages encryption key servers and clients.

Encryption/clientName/{clientName}: DELETE

Deletes the specified client.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `DELETE /scalemgmt/v2/encryption/clientName/{clientName}` request deletes the specified key client. For more information about the fields in the data structures that are returned, see the "mmkeyserv command" on page 469.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/clientName/{clientName}
```

Request headers

```
Accept: application/json
```

Parameters

Table 59. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
clientName	Name of the key client that must be deleted.	Required

Request data

No request data.

Response data

```
{
  "{
  "status": {
    "code": Return Code,
    "message": "String"
  },
  "jobs": [
    {
      "result": {
        "commands": "Array",
        "progress": "Array",
        "exitCode": Integer,
        "stderr": "Array",
        "stdout": "Array"
      },
      "request": {
        "type": "POST| GET | PUT | DELETE",
        "url": "String",
        "data": "String"
      },
      "jobId": Integer,
      "submitted": "Date and Time",
      "completed": "Date and Time",
      "runtime": Integer,
```

```
      "status": "RUNNING | CANCELLING | CANCELLED | COMPLETED | FAILED ",
      "pids": "Integer"
    }
  ]
  }
  }
```

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

"jobs"
An array that comprises details on the jobs that were run.

"result":
Specifies the job results.

"commands": *commands executed*
Specifies the commands that were issued for the job.

"progress":"*Job progress* "
Specifies the progress of the job.

"exitCode":"*exit codes*"
Specifies the exit code of the command. A value zero indicates success while any other zero indicates failure.

"stderr":"*CLI messages*"
Specifies the CLI messages received from stderr.

"stdout":"*CLI messages*"
Specifies the CLI messages from stdout.

result
Specifies the job results.

"type": "*GET | POST | PUT | DELETE*
Specifies the HTTP request type.

"URL" : "*URL*"
Specifies the URL through which the job is submitted.

"Data" : "*request data*"
Specifies the request data. Optional.

"jobId": "*ID*",
Specifies the unique ID of the job.

"submitted": "*Date and Time*"
Specifies the date and time at which the job was submitted.

"completed": "*Date and Time*"
Specifies the date and time at which the job was completed.

"runtime": "*Duration*"
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"pids":"Process ID"
Specifies the process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to delete a key client:

Request data:

```
 curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/encryption/clientName/myclient1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000002,
    "status" : "COMPLETED",
    "submitted" : "2021-01-30 18:01:45,173",
    "completed" : "2021-01-30 18:01:45,737",
    "runtime" : 566,
    "request" : {
      "type" : "DELETE",
      "url" : "scalemgmt/v2/encryption/clientName/myclient1"
      "data": " "
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmkeyserv client delete 'myclient1' " ],
      "stdout" : [ " " ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Related reference

“mmkeyserv command” on page 469
Manages encryption key servers and clients.

Encryption/deleteTenant: DELETE

Deletes a tenant from the RKM server.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `DELETE /scalemgmt/v2/encryption/deleteTenant` request deletes a tenant from the remote key manager (RKM) server. A tenant is an IBM Security Key Lifecycle Manager device group that comprises the encryption keys for registered key clients. For more information about the fields in the data structures that are returned, see the "mmkeyserv command" on page 469.

Request URL

```
https://<IP address or hostname of API server>:<port>/scalemgmt/v2/encryption/ deleteTenant
```

Request headers

```
Accept: application/json
```

Request data

```
{
  "tenantName": "Tenant name",
  "serverName": "Server name",
  "passwordFile": "Password file"
}
```

The details of the parameters are given in the following list.

"tenantName": "*Tenant name*"
: The name of the tenant that is deleted.

"serverName": "*Server name*"
: The hostname or the IP address of the RKM server to which the tenant belongs.

"passwordFile": "*Password file*"
: The password file that contains a password for accessing the RKM server. If you do not provide a password, you are prompted for a password when the request is sent. A password must be 1 - 20 characters in length. For more information, see the **mmkeyserv** command in the IBM Spectrum Scale documentation.

Response data

```
{
  "{
  "status": {
    "code": Return Code,
    "message": "String"
  },
  "jobs": [
    {
      "result": {
        "commands": "Array",
        "progress": "Array",
        "exitCode": Integer,
        "stderr": "Array",
        "stdout": "Array"
      },
```

```
      "request": {
        "type": "POST| GET | PUT | DELETE",
        "url": "String",
        "data": "String"
      },
      "jobId": Integer,
      "submitted": "Date and Time",
      "completed": "Date and Time",
      "runtime": Integer,
      "status": "RUNNING | CANCELLING | CANCELLED | COMPLETED | FAILED ",
      "pids": "Integer"
    }
  ]
}   }
  ]
}
```

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

"jobs"
An array that comprises details on the jobs that were run.

"result":
Specifies the job results.

"commands": *Commands executed*
Specifies the commands that were issued for the job.

"progress":"*Job progress* "
Specifies the progress of the job.

"exitCode":"*Exit codes*"
Specifies the exit code of the command. Zero indicates success while any value other than zero indicates failure.

"stderr":"*CLI messages*"
Specifies the CLI messages received from stderr.

"stdout":"*CLI messages*"
Specifies the CLI messages from stdout.

result
Specifies the job results.

"type": "*GET | POST | PUT| DELETE*
Specifies the HTTP request type.

"URL" : "*URL*"
Specifies the URL through which the job is submitted.

"Data" : "*Request data*"
Specifies the request data. Optional.

"jobId": "*ID*",
Specifies the unique ID of the job.

"submitted": "*Date and Time*"
Specifies the date and time at which the job was submitted.

"completed": "*Date and Time*"
Specifies the date and time at which the job was completed.

"runtime": "*Duration*"
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"pids":"Process ID"
Specifies the process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to delete a tenant:

Request data:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json' -d '{ \
  "tenantName": "devG1",
 "serverName": "sklm11.fyre.ibm.com",
 "passwordFile": "/tmp/password",
 'https://198.51.100.1:443/scalemgmt/v2/encryption/deleteTenant'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000002,
    "status" : "COMPLETED",
    "submitted" : "2021-01-30 18:01:45,173",
    "completed" : "2021-01-30 18:01:45,737",
    "runtime" : 566,
    "request" : {
      "type" : "DELETE",
      "url" : "scalemgmt/v2/encryption/deleteTenant"
      "data": " "
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmkeyserv key delete --server 'sklm11.fyre.ibm.com' [--server-pwd '/tmp/
password']
   {--all --tenant 'devG1' | --file '1'}" ],
      "stdout" : [ " " ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Related reference

“mmkeyserv command” on page 469
Manages encryption key servers and clients.

Encryption/serverName/{serverName}: DELETE

Deletes the specified sever.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `DELETE /scalemgmt/v2/encryption/serverName/{serverName}` request deletes the specified remote key manager (RKM) server. For more information about the fields in the data structures that are returned, see the "mmkeyserv command" on page 469.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/encryption/serverName/
{serverName}
```

where

{serverName}
Specifies the RKM server that must be deleted.

Request headers

```
Accept: application/json
```

Request data

No request data.

Response data

```
{
  "{
  "status": {
    "code": Return Code,
    "message": "String"
  },
  "jobs": [
    {
      "result": {
        "commands": "Array",
        "progress": "Array",
        "exitCode": Integer,
        "stderr": "Array",
        "stdout": "Array"
      },
      "request": {
        "type": "POST| GET | PUT | DELETE",
        "url": "String",
        "data": "String"
      },
      "jobId": Integer,
      "submitted": "Date and Time",
      "completed": "Date and Time",
      "runtime": Integer,
      "status": "RUNNING | CANCELLING | CANCELLED | COMPLETED | FAILED ",
      "pids": "Integer"
    }
  ]
}    }
```

```
  ]
}
```

"status":
Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

"jobs"
An array that comprises details on the jobs that were run.

"result":
Specifies the job results.

"commands": *commands executed*
Specifies the commands that were issued for the job.

"progress":"*Job progress* "
Specifies the progress of the job.

"exitCode":"*exit codes*"
Specifies the exit code of the command. A value zero indicates success while any other zero indicates failure.

"stderr":"*CLI messages*"
Specifies the CLI messages received from stderr.

"stdout":"*CLI messages*"
Specifies the CLI messages from stdout.

result
Specifies the job results.

"type": "*GET | POST | PUT| DELETE*
Specifies the HTTP request type.

"URL" : "*URL*"
Specifies the URL through which the job is submitted.

"Data" : "*request data*"
Specifies the request data. Optional.

"jobId": "*ID*",
Specifies the unique ID of the job.

"submitted": "*Date and Time*"
Specifies the date and time at which the job was submitted.

"completed": "*Date and Time*"
Specifies the date and time at which the job was completed.

"runtime": "*Duration*"
Specifies the duration for which the job ran.

"status":"RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"pids":"Process ID"
Specifies the process IDs of all the active sub processes that manage the job.

Examples

The following example shows how to delete the specified RKM server:

Request data:

```
 curl -k -u admin:admin001 -X DELETE --header 'accept:application/json''https://
198.51.100.1:443/scalemgmt/v2/encryption/serverName/sklm11.fyre.ibm.com'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000002,
    "status" : "COMPLETED",
    "submitted" : "2021-01-30 18:01:45,173",
    "completed" : "2021-01-30 18:01:45,737",
    "runtime" : 566,
    "request" : {
      "type" : "DELETE",
      "url" : "scalemgmt/v2/encryption/serverName/sklm11.fyre.ibm.com"
      "data": " "
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmkeyserv server delete sklm11.fyre.ibm.com" ],
      "stdout" : [ " " ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Related reference

“mmkeyserv command” on page 469
Manages encryption key servers and clients.

Filesystems/{filesystemName}/suspend: PUT

Suspends the processing of I/O requests for a file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `filesystems/{filesystemName}/suspend` request temporarily suspends processing of all I/O requests of the file system. For more information about the fields in the data structures that are returned, see the “mmfsctl command” on page 424.

Request URL

```
https://<IP address or host name of API server>:<port>scalemgmt/v2/filesystems/{filesystemName}/
suspend
```

where

`filesystemName`
: Specifies the filesystem that is temporarily suspended from receiving or sending I/O requests. Required.

Request headers

```
Accept: application/json
```

Request data

No request data.

Response data

```
{
  "jobs": [
    {
      "jobId": ID of the job,
      "status": "Job status",
      "submitted": "Date and time ",
      "completed": "Date and time ",
      "runtime": Duration,
      "request": {
        "type": "Request Type",
        "url": "Resource URL"
      },
      "result": {},
      "pids": []
    }
  ],
  "status": {
    "code": return status code,
    "message": "Return message."
  }
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
: An array of elements that describe jobs. Each element describes one job.

"jobId": "*ID*",
Specifies the unique ID of the job.

"submitted": "*Time*"
Specifies the time at which the job was submitted.

"completed": "*Time*"
Specifies the time at which the job was completed.

"runtime": "*Time*"
Specifies the duration for which the job ran.

"status": "RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"result"
Array of commands that are run in this job.

"pids": *list*
A list of pids for this job.

"request"

"type": "{GET | POST | PUT | DELETE}"
Specifies the HTTP request type.

"url": "*URL*"
Specifies the URL through which the job is submitted.

"status":
Return status.

"message": "*ReturnMessage*",
Specifies the return message.

"code": *ReturnCode*
Specifies the return code.

Examples

The following example suspends the file system *FileS1*.

Request data:

```
curl -k -X PUT --header 'Content-Type: application/json' --header 'Accept: application/json'
--header 'Authorization: Basic YWRtaW46VHJhY2VAMjAyMQ==' 'https://198.51.100.1:443/scalemgmt/v2/
filesystems/FileS1/suspend'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs": [
    {
      "jobId": 1000000000001,
      "status": "RUNNING",
      "submitted": "2021-03-03 07:41:04,436",
      "completed": "N/A",
      "runtime": 15,
      "request": {
        "type": "PUT",
        "url": "/scalemgmt/v2/filesystems/FileS1/suspend"
      },
      "result": {},
      "pids": []
    }
  ],
  "status": {
    "code": 202,
    "message": "The request was accepted for processing."
```

```
    }
}
```

Related reference

“mmfsctl command” on page 424
Issues a file system control request.

Filesystems/{filesystemName}/resume: PUT

Resumes the processing of I/O requests for a file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `filesystems/{filesystemName}/resume` request resumes the processing of all suspended I/O requests for a specific file system. For more information about the fields in the data structures that are returned, see the "mmfsctl command" on page 424.

Request URL

```
https://<IP address or host name of API server>:<port>scalemgmt/v2/filesystems/{filesystemName}/
resume
```

where

filesystemName
Specifies the filesystem for which the processing of I/O requests is resumed. Required.

Request headers

```
Accept: application/json
```

Request data

No request data.

Response data

```
{
  "jobs": [
    {
      "jobId": ID of the job,
      "status": "Job status",
      "submitted": "Date and time",
      "completed": "Date and time",
      "runtime": Duration,
      "request": {
        "type": "Request Type",
        "url": "Resource URL"
      },
      "result": {},
      "pids": []
    }
  ],
  "status": {
    "code": return status code,
    "message": "Return message."
  }
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"jobId": "*ID*",
Specifies the unique ID of the job.

"submitted": "*Time*"
Specifies the date and time at which the job was submitted.

"completed": "*Time*"
Specifies the date and time at which the job was completed.

"runtime": "*Time*"
Specifies the duration for which the job ran.

"status": "RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"result"
Array of commands that are run in this job.

"pids": *list*
A list of pids for this job.

"request"

"type": "{GET | POST | PUT | DELETE}"
Specifies the HTTP request type.

"url": "*URL*"
Specifies the URL through which the job is submitted.

"status":
Return status.

"message": "*ReturnMessage*",
Specifies the return message.

"code": *ReturnCode*
Specifies the return code.

Examples

The following example resumes the processing of I/O requests for the file system *FS1*.

Request data:

```
curl -k -X PUT --header 'Content-Type: application/json' --header 'Accept: application/json'
--header 'Authorization: Basic YWRtaW46VHJhY2VAMjAyMQ==' 'https://198.51.100.1:443/scalemgmt/v2/
filesystems/FS1/resume'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs": [
    {
      "jobId": 1000000000002,
      "status": "RUNNING",
      "submitted": "2021-03-03 07:41:55,792",
      "completed": "N/A",
      "runtime": 2,
      "request": {
        "type": "PUT",
        "url": "/scalemgmt/v2/filesystems/FS1/resume"
      },
      "result": {},
      "pids": []
    }
  ],
  "status": {
    "code": 202,
    "message": "The request was accepted for processing."
  }
}
```

Related reference

“mmfsctl command” on page 424
Issues a file system control request.

Health/config/{interval}: PUT

Configures the interval at which the node health data needs to be collected.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `scalemgmt/v2/health/config/{interval}` request configures the interval at which the health data for the nodes and services needs to be collected. For more information about the fields in the data structures that are returned, see the “mmhealth command” on page 441.

Request URL

```
https://<IP address or host name of API server>:<port> scalemgmt/v2/health/config/{interval}
```

where

`interval`
Specifies the monitoring interval for the whole cluster. The interval values are shown in the following list. Required.

OFF
The monitoring is disabled for the whole cluster.

LOW
The monitoring time is equivalent to the default monitoring multiplied by10 seconds.

MEDIUM
The monitoring time is equivalent to the default monitoring multiplied by 5 seconds.

DEFAULT
Monitoring is set for every 15 - 30 seconds based on the service that is being monitored.

HIGH
Monitoring time is equivalent to the default monitoring time divided by 2 seconds.

Request headers

```
Accept: application/json
```

Request data

No request data.

Response data

```
{
  "jobs": [
    {
      "jobId": ID of the job,
      "status": "Job status",
      "submitted": "Time",
      "completed": "Time",
      "runtime": Duration,
      "request": {
        "type": "Request Type",
        "url": "Resource URL"
      },
      "result": {},
      "pids": []
```

```
    }
  ],
  "status": {
    "code": return status code,
    "message": "Return message."
  }
}
```

For more information about the fields in the following data structures, see the link at the end of the topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"jobId": "*ID*",
Specifies the unique ID of the job.

"submitted": "*Time*"
Specifies the time at which the job was submitted.

"completed": "*Time*"
Specifies the time at which the job was completed.

"runtime": "*Time*"
Specifies the duration for which the job ran.

"status": "RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"result"
The request results.

"pids": *list*
A list of pids for this job.

"request"

"type": "{GET | POST | PUT | DELETE}"
Specifies the HTTP request type.

"url": "*URL*"
The URL through which the job is submitted.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example sets the monitoring interval to *LOW*.

Request data:

```
curl -k -X PUT --header 'Content-Type: application/json' --header 'Accept: application/json'
--header 'Authorization: Basic YWRtaW46YWRtaW4wMDE=' ' https://198.51.100.1:443/scalemgmt/v2/
health/config/LOW'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs": [
    {
      "jobId": 1000000000001,,
```

```
      "status": "RUNNING",
      "submitted": "2021-04-26 03:17:14,944",
      "completed": "N/A",
      "runtime": 4,
      "request": {
        "type": "PUT",
        "url": "/scalemgmt/v2/health/config/LOW"

      },
      "result": {},
      "pids": []
    }
  ],
  "status": {
    "code": 202,
    "message": "The request was accepted for processing."
  }
}
```

Related reference

“mmhealth command” on page 441
Monitors health status of nodes.

Filesystem/{filesystemName}/maintenancemode/{maintenanceModeoption}: PUT

Enables the maintenance mode for a specific file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `filesystem/{filesystemName}/maintenanceMode/{maintenanceModeoption}` request enables the maintenance mode for a specified file system when it is unmounted. The query **`waitRequired`** is added to indicate that a waiting time is needed to allow the file system to unmount before the request is sent. For more information about the fields in the data structures that are returned, see the "mmchfs command" on page 232.

Request URL

```
https://<IP address or host name of API server>:<port> scalemgmt/v2/filesystem/{fileSystemName}/
maintenanceMode/{maintenanceModeoption}
```

Request headers

```
Accept: application/json
```

Request data

No request data.

Parameters

Table 60. List of parameters

Parameter name	Description and applicable keywords	Required/Optional
filesystemName	Specifies the file system for which the maintenance is being enabled.	Required
maintenanceModeoption	The option that indicates whether the maintenance mode is being enabled.	Required
waitRequired	Specifies whether the query must wait for the file system to be unmounted before the maintenance mode is turned on.	Optional

Response data

```
{
  "jobs": [
    {
      "jobId": ID of the job,
      "status": "Job status",
      "submitted": "Time",
```

```
      "completed": "Time",
      "runtime": Duration ,
      "request": {
        "type": "Request Type",
        "url": "Resource URL"
      },
      "result": {},
      "pids": []
    }
  ],
  "status": {
    "code": return status code,
    "message": "Return message."
  }
}
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"jobId": "*ID*",
Specifies the unique ID of the job.

"submitted": "*Time*"
Specifies the date and time when the job was submitted.

"completed": "*Time*"
Specifies the date and time when the job was completed.

"runtime": "*Time*"
Specifies the duration for which the job ran.

"status": "RUNNING | COMPLETED | FAILED"
Specifies the status of the job.

"result"
An array of commands that are run in this job.

"pids": *list*
A list of pids for this job.

"request"

"type": "{GET | POST | PUT | DELETE}"
Specifies the HTTP request type.

"url": "*URL*"
Specifies the URL through which the job is submitted.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example enables the maintenance mode for the file system *FS1*.

Request data:

```
curl -k -X PUT --header 'Content-Type: application/json' --header 'Accept: application/json'
--header 'Authorization: Basic YWRtaW46YWRtaW4wMDE=' 'https://198.51.100.1:443/scalemgmt/v2/
filesystems/FS1/maintenanceMode/yes?waitRequired=wait'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs": [
    {
      "jobId": 1000000000004,
      "status": "RUNNING",
      "submitted": "2021-04-26 09:16:04,355",
      "completed": "N/A",
      "runtime": 3,
      "request": {
        "type": "PUT",
        "url": "/scalemgmt/v2/filesystems/FS1/maintenanceMode/yes"
      },
      "result": {},
      "pids": []
    }
  ],
  "status": {
    "code": 202,
    "message": "The request was accepted for processing."
  }
}
```

Related reference

“mmchfs command” on page 232
Changes the attributes of a GPFS file system.

Diagnostic/snap/{snapPath}: DELETE

Deletes the snap file comprising the diagnostic data that is collected by the **gpfs.snap** command.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `DELETE diagnostic/snap/{snapPath}` request deletes the specified snap file. For more information about the fields in the data structures that are returned, see "gpfs.snap command" on page 8.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/diagnostic/snap/{snapPath}
```

where

diagnostic/snap/*{snapPath}*
: Specifies the snap file to be deleted. Required.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 61. List of parameters

Parameter name	Description and applicable keywords	Required/optional
snapPath	The snap file path provided as a URL	Required.

Request data

No request data.

Response data

```
{
   "status" : {
    "code" : The Return Code,
    "message" : "The ReturnMessage."
  }
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
: Return status.

  "code": "*ReturnCode*"
  : The return code.

  "message": "*ReturnMessage*",
  : The return message.

Examples

The following example shows how to delete a snap file.

Request data:

```
curl -k -u 'username:password' -X DELETE --header 'Accept: application/json' --header
'Authorization: Basic YWRtaW46YWRtaW4wMDE=' 'https://198.51.100.1:443/scalemgmt/v2/diagnostic/
snap/%2Ftmp%2Fgpfs.snapOut%2F3%2Fall.20210128174912.616684.tar'>/tmp/snap.tar'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
   "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Related reference

“gpfs.snap command” on page 8
Creates an informational system snapshot at a single point in time. This system snapshot consists of information such as cluster configuration, disk configuration, network configuration, network status, GPFS logs, dumps, and traces.

Filesystems: GET

Gets information about file systems.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET filesystems` request gets information about file systems in the cluster. For more information about the fields in the data structures that are returned, see the topics “mmcrfs command” on page 318, “mmchfs command” on page 232, and “mmlsfs command” on page 506.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems
```

where

filesystems
Specifies file systems as the resource of the GET call.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 62. List of parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Request data

No request data.

Response data

```
{
   "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
    },
    "paging":
    {
    "next": "URL"
    },
```

```
   filesystems: [
      {
         "oid":"Internal ID",
         "uuid":"ID",
         "name":"Name",
         "version":"Version",
         "type":"Type",
         "createTime":"DateTime",
         "block"
             {
              "pools":"Pools",
              "disks":"List of disks",
              "blockSize":Block size,
              "metaDataBlockSize":Metadata block size,
              "indirectBlockSize":Indirect block size,
              "minFragmentSize":Minimum fragment size,
              "inodeSize":Inode size,
              "logfileSize":Log file size,
              "writeCacheThreshold":Threshold
            },
         "mount"
             {
              "mountPoint":Path
              "automaticMountOption":"{yes | no | automount }",
              "additionalMountOptions":"Options",
              "mountPriority":Priority,
              "driveLetter":Drive letter,
              "remoteDeviceName":Device name,
              "readOnly":Read-only,
             }
         "replication"
             {
              "defaultMetadataReplicas":Number,
              "maxMetadataReplicas":Number,
              "defaultDataReplicas":Number,
              "maxDataReplicas":Number,
              "strictReplication":"{no | whenpossible | always }",
              "logReplicas":Number,
             }
         "quota"
             (
              "quotasAccountingEnabled":"[user][;group][;fileset]"
              "quotasEnforced":"[user][;group][;fileset]"
              "defaultQuotasEnabled":"[user][;group][;fileset]"
              "perfilesetQuotas":"{yes | no}",
              "filesetdfEnabled":"{yes | no}",
             }
         "settings"
             (
              "blockAllocationType":"{scatter | cluster}",
              "fileLockingSemantics":"{nfs4 | posix}",
              "numNodes":Number,
              "exactMTime":"{true | false}",
              "suppressATime":"{yes | no}",
              "fastEAEnabled":"{yes | no}",
              "encryption":"{yes | no}",
              "maxNumberOfInodes":"Number",
              "is4KAligned":"{yes | no}",
              "rapidRepairEnabled":"{yes | no}",
              "stripeMethod":"{yes | no}",
              "stripedLogs":"{yes | no}",
              "ACLSemantics":"{posix | afs4 | all }",
              "DMAPIEnabled":"{yes | no}",
             }
            "fileAuditLogConfig"
             (
              "auditFilesetDeviceName":"name",
              "auditFilesetName":"Audit fileset name",
              "auditRetention": "Retention period",
              "eventTypes": "List of events to audit",
              "topicGenNum":"Generation number",
             }
          }
   ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": "*ReturnCode*"
The return code.

"filesystems":
An array of elements that describe file systems. Each element describes one file system.

"oid":"*Internal ID*",
Internal identifier that is used for paging.

"uuid":"*ID*"
The UUID of the file system.

"name":"*Name*"
Name of the file system.

"version":"*Version*"
File system version.

"type":"*local | remote*"
File system type.

"createTime":"*DateTime*"
Creation time.

"block"

"pools":"*Pools*"
List of pools of this file system.

"disks":"*List of disks*"
A semicolon-separated list of the disks that are included in the file system.

"blockSize":"*Block size*"
The block size of the disks in the storage pool.

"metaDataBlockSize":"*Metadata block size*"
Block size of metadata pool.

"indirectBlockSize":"*Indirect block size*"
Indirect block size in bytes.

"minFragmentSize":"*Minimum fragment size*"
Minimum fragment size in bytes.

"inodeSize":"*Inode size*"
Inode size in bytes.

"logfileSize":"*Log file size*"
The size of the internal log files in bytes.

"writeCacheThreshold":"*Threshold*"
Specifies the maximum length (in bytes) of write requests that are initially buffered in the highly-available write cache before being written back to primary storage.

"mount"

"mountPoint":"*Path*"
Default mount point.

"automaticMountOption":"{yes | no | automount }"
Indicates when the file system is to be mounted: *yes* when GPFS daemon starts, *no* to manual mount, *automount* when the file system is first accessed.

"additionalMountOptions":"*Options*"
The mount options to pass to the mount command while mounting the file system.

"mountPriority":"*Priority*"
Controls the order in which the individual file systems are mounted at daemon startup or when one of the all keywords is specified on the **mmmount** command.

"driveLetter":"*Drive letter*"
Drive letter when mounted on windows nodes.

"remoteDeviceName":"*Device name*"
Device name on the owning cluster of this cross-cluster mounted file system.

"readOnly":"*Read-only*"
Read-only file system.

"replication"

"defaultMetadataReplicas":"*Number*"
Default number of metadata replicas.

"maxMetadataReplicas":"*Number*"
Maximum number of metadata replicas.

"defaultDataReplicas":"*Number*"
Default number of data replicas.

"maxDataReplicas":"*Number*"
Maximum number of data replicas.

"strictReplication":"{no | whenpossible | always }"
Whether strict replication is to be enforced. 'no': not enforced, 'whenpossible': enforced if at least two failure groups present, 'always': always enforced.

"logReplicas":"*Number*"
Number of log replicas.

"quotas"

"quotasAccountingEnabled":"[user][;group][;fileset]"
The types of quotas for which accounting is enabled. Accounting means that usage is tracked against quotas. If action needs to be taken when quotas are exceeded, quotas must be also enforced.

"quotasEnforced":"[user][;group][;fileset]"
Quotas that are enforced for any of 'user', 'group' and 'fileset'.

"defaultQuotasEnabled":"[user][;group][;fileset]"
Default quotas that are enabled.

"perfilesetQuotas":"{yes | no}"
Specifies whether the scope of user and group quota limit checks on the individual fileset level rather than the entire file system.

"filesetdfEnabled":"{yes | no}"
Specifies whether the quotas are enforced for a fileset, the numbers reported by the **df** command are based on the quotas for the fileset rather than the entire file system.

"settings"

"blockAllocationType":"{scatter | cluster}"
Block allocation type.

"fileLockingSemantics":"{nfs4 | posix}"
File locking semantics in effect.

"numNodes":*Number*,
Estimated number of nodes that will mount file system.

"exactMTime":"{true | false}"
`true` reports exact mtime values and `false` mtime is only periodically updated.

"suppressATime":"{yes | no | realtime}"
Suppress the periodic updating of the value of atime as reported by stat calls.

"fastEAEnabled":"{yes | no}"
Specifies whether fast extended attributes are enabled.

"encryption":"{yes | no}"
Specifies whether encryption is enabled.

"maxNumberOfInodes":"*Number*"
Specifies the maximum number of files that can be created inside the inode space of the root fileset of this file system.

"is4KAligned":"{yes | no}"
Whether the file system is formatted to be 4K aligned.

"rapidRepairEnabled":"{yes | no}"
Whether to keep track of incomplete replication on an individual file block basis as opposed to the entire file.

"stripeMethod":"{yes | no}"
Algorithm that is used for striping data.

"stripedLogs":"{yes | no}"
Metadata logs are striped across available metadata disks.

"ACLSemantics":"{posix | afs4 | all }"
ACL semantics supported by this file system.

"DMAPIEnabled":"{yes | no}"
Whether Data Management API is enabled on this file system.

"fileAuditLogConfig"

" "auditFilesetDeviceName":"name"
The name of the file system where the filesets that contains the audit log records reside.

"auditFilesetName":"Audit fileset name"
The name of the fileset where the audit log records are stored.

"auditRetention": "Retention period"
Number of days for which the audit log records are kept.

"eventTypes": "List of event types to audit"
Specifies the list of event types to be included in the file audit log.

"topicGenNum":"Generation number"
The topic generation number.

Examples

The following example gets information about file systems in the cluster.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "filesystems" : [ {
    "name" : "gpfs0"
  }, {
    "name" : "objfs"
  }, {
    "name" : "gpfs1"
  } ],
  "status" : {
```

```
    "code" : 200,
    "message" : "The request finished successfully"
  }
}
```

Using the field parameter ":all:" returns entire details of the file systems that are configured in the system. The **fileystems** array returns file system objects in the following example. Each object contains details about one file system:

```
{
  "status": {
    "code": 200,
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/filesystems/gpfs0/filesets?lastId=10001",
    "fields": "period,restrict,sensorName",
    "filter": "usedInodes>100,maxInodes>1024",
    "baseUrl": "/scalemgmt/v2/perfmon/sensor/config",
    "lastId": 10001
  },
  "filesystems": [
    {
      "oid": 2,
      "uuid": "0A00641F:58982753",
      "name": "gpfs0",
      "version": "17.00",
      "type": "local",
      "createTime": "Mon Feb 06 08:35:47 2017",
      "block": {
        "pools": "system;data",
        "disks": "disk1;disk2",
        "blockSize": 262144,
        "metaDataBlockSize": 262144,
        "indirectBlockSize": 16384,
        "minFragmentSize": 8192,
        "inodeSize": 4096,
        "logfileSize": 4194304,
        "writeCacheThreshold": 0
      },
      "mount": {
        "mountPoint": "/mnt/gpfs0",
        "automaticMountOption": "yes",
        "additionalMountOptions": "none",
        "mountPriority": 0,
        "driveLetter": "F",
        "remoteDeviceName": "gpfs0",
        "readOnly": false
      },
      "replication": {
        "defaultMetadataReplicas": 1,
        "maxMetadataReplicas": 2,
        "defaultDataReplicas": 1,
        "maxDataReplicas": 2,
        "strictReplication": "whenpossible",
        "logReplicas": 0
      },
      "quota": {
        "quotasAccountingEnabled": "user;group;fileset",
        "quotasEnforced": "user;group;fileset",
        "defaultQuotasEnabled": "none",
        "perfilesetQuotas": true,
        "filesetdfEnabled": false
      },
      "settings": {
        "blockAllocationType": "cluster",
        "fileLockingSemantics": "nfs4",
        "aclSemantics": "nfs4",
        "numNodes": 32,
        "dmapiEnabled": true,
        "exactMTime": true,
        "suppressATime": "no",
        "fastEAEnabled": true,
        "encryption": false,
        "maxNumberOfInodes": 3211520,
        "is4KAligned": true,
        "rapidRepairEnabled": true,
        "stripeMethod": "round-robin",
        "stripedLogs": true,
        "fileAuditLogEnabled": true
```

```
      },
      "fileAuditLogConfig": {
        "auditFilesetDeviceName": "logFilesystem",
        "auditFilesetName": "logFileset1",
        "auditRetention": 365,
        "topicGenNum": 123,
        "eventTypes":
"ACLCHANGE,CLOSE,CREATE,DESTROY,GPFSATTRCHANGE,OPEN,RENAME,RMDIR,UNLINK,XATTRCHANGE"
      }
    }
  ]
}
```

“mmcrfs command” on page 318
Creates a GPFS file system.

“mmchfs command” on page 232
Changes the attributes of a GPFS file system.

“mmlsfs command” on page 506
Displays file system attributes.

Filesystems/{filesystemName}: GET

Gets information about a file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET filesystems/filesystemName` request gets information about a particular file system. For more information about the fields in the data structures that are returned, see the topics "mmcrfs command" on page 318, "mmchfs command" on page 232, and "mmlsfs command" on page 506.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/fileSystemName
```

where

filesystems
Specifies file system as the resource of the GET call. Required.

FileSystemName
The file system about which you want to get information. Required.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 63. List of parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	

Request data

No request data.

Response data

```
{
   "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
    },
    "paging":
    {
    "next": "URL"
    },
   filesystems: [
      {
         "oid":"Internal ID",
         "uuid":"ID",
         "name":"Name",
         "version":"Version",
         "type":"Type",
         "createTime":"DateTime",
         "block"
             {
              "pools":"Pools",
              "disks":"List of disks",
              "blockSize":Block size,
              "metaDatBlockSize":Metadata block size,
              "indirectBlockSize":Indirect block size,
              "minFragmentSize":Minimum fragment size,
              "inodeSize":Inode size,
              "logfileSize":Log file size,
              "writeCacheThreshold":Threshold
            },
         "mount"
             (
              "mountPoint":Path
              "automaticMountOption":"{yes | no | automount }",
              "additionalMountOptions":"Options",
              "mountPriority":Priority,
              "driveLetter":Drive letter,
              "remoteDeviceName":Device name,
              "readOnly":Read-only,
             }
         "replication"
             {
              "defaultMetadataReplicas":Number,
              "maxMetadataReplicas":Number,
              "defaultDataReplicas":Number,
              "maxDataReplicas":Number,
              "strictReplication":"{no | whenpossible | always }",
              "logReplicas":Number,
             )
         "quota"
             (
              "quotasAccountingEnabled":"[user][;group][;fileset]"
              "quotasEnforced":"[user][;group][;fileset]"
              "defaultQuotasEnabled":"[user][;group][;fileset]"
              "perfilesetQuotas":"{yes | no}",
              "filesetdfEnabled":"{yes | no}",
             }
         "settings"
             (
              "blockAllocationType":"{scatter | cluster}",
              "fileLockingSemantics":"{nfs4 | posix}",
              "numNodes":Number,
              "exactMTime":"{true | false}",
              "suppressATime":"{yes | no}",
              "fastEAEnabled":"{yes | no}",
              "encryption":"{yes | no}",
              "maxNumberOfInodes":Number,
              "is4KAligned":"{yes | no}",
              "rapidRepairEnabled":"{yes | no}",
              "stripeMethod":"{yes | no}",
              "stripedLogs":"{yes | no}",
              "ACLSemantics":"{posix | afs4 | all }",
              "DMAPIEnabled":"{yes | no}",
       "fileAuditLogEnabled":"{yes | no}",
             }
          "fileAuditLogConfig"
             (
              "auditFilesetDeviceName":"name",
              "auditFilesetName":"Audit fileset name",
```

```
                "auditRetention": "Retention period",
                "eventTypes": "List of events to audit",
                "topicGenNum":"Generation number",
               }
            }
    ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": "*ReturnCode*"
The return code.

"filesystems":
An array of elements that describe file systems. Each element describes one file system.

"oid":"*Internal ID*",
Internal identifier that is used for paging.

"uuid":"*ID*"
The UUID of the file system.

"name":"*Name*"
Name of the file system.

"version":"*Version*"
File system version.

"type":"*local | remote*"
File system type.

"createTime":"*DateTime*"
Creation time.

"block"

"pools":"*Pools*"
List of pools of this file system.

"disks":"*List of disks*"
A semicolon-separated list of the disks that are included in the file system.

"blockSize":"*Block size*"
The block size of the disks in the storage pool.

"metaDataBlockSize":"*Metadata block size*"
Block size of metadata pool.

"indirectBlockSize":"*Indirect block size*"
Indirect block size in bytes.

"minFragmentSize":"*Minimum fragment size*"
Minimum fragment size in bytes.

"inodeSize":"*Inode size*"
Inode size in bytes.

"logfileSize":"*Log file size*"
The size of the internal log files in bytes.

"writeCacheThreshold":"*Threshold*"
Specifies the maximum length (in bytes) of write requests that are initially buffered in the highly-available write cache before being written back to primary storage.

"mount"

"mountPoint":"*Path*"
Default mount point.

"automaticMountOption":"{yes | no | automount }"
Indicates when the file system is to be mounted: *yes* when GPFS daemon starts, *no* to manual mount, *automount* when the file system is first accessed.

"additionalMountOptions":"*Options*"
The mount options to pass to the mount command while mounting the file system.

"mountPriority":"*Priority*"
Controls the order in which the individual file systems are mounted at daemon startup or when one of the all keywords is specified on the **mmmount** command.

"driveLetter":"*Drive letter*"
Drive letter when mounted on windows nodes.

"remoteDeviceName":"*Device name*"
Device name on the owning cluster of this cross-cluster mounted file system.

"readOnly":"*Read-only*"
Read-only file system.

"replication"

"defaultMetadataReplicas":"*Number*"
Default number of metadata replicas.

"maxMetadataReplicas":"*Number*"
Maximum number of metadata replicas.

"defaultDataReplicas":"*Number*"
Default number of data replicas.

"maxDataReplicas":"*Number*"
Maximum number of data replicas.

"strictReplication":"{no | whenpossible | always }"
Whether strict replication is to be enforced. 'no': not enforced, 'whenpossible': enforced if at least two failure groups present, 'always': always enforced.

"logReplicas":"*Number*"
Number of log replicas.

"quotas"

"quotasAccountingEnabled":"[user][;group][;fileset]"
The types of quotas for which accounting is enabled. Accounting means that usage is tracked against quotas. If action needs to be taken when quotas are exceeded, quotas must be also enforced.

"quotasEnforced":"[user][;group][;fileset]"
Quotas that are enforced for any of 'user', 'group' and 'fileset'.

"defaultQuotasEnabled":"[user][;group][;fileset]"
Default quotas that are enabled.

"perfilesetQuotas":"{yes | no}"
Specifies whether the scope of user and group quota limit checks on the individual fileset level rather than the entire file system.

"filesetdfEnabled":"{yes | no}"
Specifies whether the quotas are enforced for a fileset, the numbers reported by the **df** command are based on the quotas for the fileset rather than the entire file system.

"settings"

"blockAllocationType":"{scatter | cluster}"
Block allocation type.

"fileLockingSemantics":"{nfs4 | posix}"
File locking semantics in effect.

"numNodes":*Number*,
Estimated number of nodes that will mount file system.

"exactMTime":"{true | false}"
`true` reports exact mtime values and `false` mtime is only periodically updated.

"suppressATime":"{yes | no | realtime}"
Suppress the periodic updating of the value of atime as reported by stat calls.

"fastEAEnabled":"{yes | no}"
Specifies whether fast extended attributes are enabled.

"encryption":"{yes | no}"
Specifies whether encryption is enabled.

"maxNumberOfInodes":*Number*"
Specifies the maximum number of files that can be created inside the inode space of the root fileset of this file system.

"is4KAligned":"{yes | no}"
Whether the file system is formatted to be 4K aligned.

"rapidRepairEnabled":"{yes | no}"
Whether to keep track of incomplete replication on an individual file block basis as opposed to the entire file.

"stripeMethod":"{yes | no}"
Algorithm that is used for striping data.

"stripedLogs":"{yes | no}"
Metadata logs are striped across available metadata disks.

"ACLSemantics":"{posix | afs4 | all }"
ACL semantics supported by this file system.

"DMAPIEnabled":"{yes | no}"
Whether Data Management API is enabled for this file system.

"fileAuditLogEnabled":"{yes | no}"
Whether file audit logging is enabled for this file system.

"fileAuditLogConfig"

" "auditFilesetDeviceName":"name"
The name of the file system where the filesets that contains the audit log records reside.

"auditFilesetName":"Audit fileset name"
The name of the fileset where the audit log records are stored.

"auditRetention": "Retention period"
Number of days for which the audit log records are kept.

"eventTypes": "List of event types to audit"
Specifies the list of event types to be included in the file audit log.

"topicGenNum":"Generation number"
The topic generation number.

Examples

The following example gets information about file system `gpfs0`.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0?fields=:all:'
```

Using the field parameter ":all:" returns entire details of the file system.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.


```
{
  "status": {
    "code": 200,
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/filesystems/gpfs0/filesets?lastId=10001",
    "fields": "period,restrict,sensorName",
    "filter": "usedInodes>100,maxInodes>1024",
    "baseUrl": "/scalemgmt/v2/perfmon/sensor/config",
    "lastId": 10001
  },
  "filesystems": [
    {
      "oid": 2,
      "uuid": "0A00641F:58982753",
      "name": "gpfs0",
      "version": "17.00",
      "type": "local",
      "createTime": "Mon Feb 06 08:35:47 2017",
      "block": {
        "pools": "system;data",
        "disks": "disk1;disk2",
        "blockSize": 262144,
        "metaDataBlockSize": 262144,
        "indirectBlockSize": 16384,
        "minFragmentSize": 8192,
        "inodeSize": 4096,
        "logfileSize": 4194304,
        "writeCacheThreshold": 0
      },
      "mount": {
        "mountPoint": "/mnt/gpfs0",
        "automaticMountOption": "yes",
        "additionalMountOptions": "none",
        "mountPriority": 0,
        "driveLetter": "F",
        "remoteDeviceName": "gpfs0",
        "readOnly": false
      },
      "replication": {
        "defaultMetadataReplicas": 1,
        "maxMetadataReplicas": 2,
        "defaultDataReplicas": 1,
        "maxDataReplicas": 2,
        "strictReplication": "whenpossible",
        "logReplicas": 0
      },
      "quota": {
        "quotasAccountingEnabled": "user;group;fileset",
        "quotasEnforced": "user;group;fileset",
        "defaultQuotasEnabled": "none",
        "perfilesetQuotas": true,
        "filesetdfEnabled": false
      },
      "settings": {
        "blockAllocationType": "cluster",
        "fileLockingSemantics": "nfs4",
        "aclSemantics": "nfs4",
        "numNodes": 32,
        "dmapiEnabled": true,
        "exactMTime": true,
        "suppressATime": "no",
        "fastEAEnabled": true,
        "encryption": false,
        "maxNumberOfInodes": 3211520,
        "is4KAligned": true,
        "rapidRepairEnabled": true,
        "stripeMethod": "round-robin",
        "stripedLogs": true,
        "fileAuditLogEnabled": true
      },
      "fileAuditLogConfig": {
        "auditFilesetDeviceName": "logFilesystem",
```

```
        "auditFilesetName": "logFileset1",
        "auditRetention": 365,
        "topicGenNum": 123,
        "eventTypes":
"ACLCHANGE,CLOSE,CREATE,DESTROY,GPFSATTRCHANGE,OPEN,RENAME,RMDIR,UNLINK,XATTRCHANGE"
      }
    }
  ]
}
```

“mmcrfs command” on page 318
Creates a GPFS file system.

“mmchfs command” on page 232
Changes the attributes of a GPFS file system.

“mmlsfs command” on page 506
Displays file system attributes.

Filesystems/{filesystemName}/acl/{path}: GET

Gets information about the access control list (ACL) set for a file or directory.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET filesystems/filesystemName/acl/path` request gets information about ACLs set for files or directories within a particular file system. For more information about the fields in the data structures that are returned, see the topics "mmgetacl command" on page 428 and "mmputacl command" on page 617.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/
FileSystemName/acl/path
```

where

filesystems/filesystemName
The file system to which the file or directory belongs. Required.

acl/*path*
The path of the file or directory about which you want to get the ACL information. Required.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 64. List of parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	Name of the file system.	Required.
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
path	The file path relative to file system's mount point. The path of the file or directory is specified with forward slashes (/). For example: `mnt/gpfs0/rest01`. The forward slashes in the path are encoded to "%2F" in the request URL.	Required.

Request data

No request data.

Response data

```
{
  "status": {
      "code":"ReturnCode,
      "message":"ReturnMessage"
   },
   acl:
      {
         "type":"{NFSv4} ",
         "entries": [
                "type":"{allow | deny | alarm | audit }",
                "who":"User or group",
                "permissions":"Access permissions",
                "flags":"fFags",
          ],

}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"acl":
An array of elements that describe ACL.

"type":"*NFSv4*"
Type of the ACL.

"entries":"*Access control entries*"

"type":"allow | deny | alarm | audit"
Type of the entry.

"who":" special:owner@ | special:group@ | special:everyone@ | user:{name} | group:{name}"
The name of the user or group of users for which the ACL is applicable.

"permission":"(r) read | (w) write | (m) mkdir, | (x) execute | (d) delete | (D) delete child | (a) read attr | (A) write attr(n) read named | (N) write Named | (c) read acl | (C) write acl | (o) change owner| (s) synchronize "
The access permissions.

"flags":"(f) file inherit | (d) dir inherit | (i) inherit only | (I) inherited | (S) successful access | (F) failed access"
Special flags and inheritance definition.

Examples

The following example gets ACL information for the file system `gpfs0` and path `/rest_fset`.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/acl/%2Frest_fset'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "acl": {
    "type": "NFSv4",
    "entries": [
      {
        "type": "allow",
        "who": "user:testuser",
        "permissions": "rxancs",
        "flags": "fd"
      }
    ]
  }
}
```

“mmgetacl command” on page 428
Displays the GPFS access control list of a file or directory.

“mmputacl command” on page 617
Sets the GPFS access control list for the specified file or directory.

Filesystems/{filesystemName}/acl/{path}: PUT

Sets access control list (ACL) for a file or directory. You can set only NFSv4 ACLs. The POSIX ACLs are not supported.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `filesystems/filesystemName/acl/path` request sets ACL for files or directories within a particular file system. For more information about the fields in the data structures that are returned, see the topics “mmgetacl command” on page 428 and “mmputacl command” on page 617.

Note: Only the users with *dataaccess* role can set ACL for a file or directory.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/
FileSystemName/acl/path
```

where

filesystems/*filesystemName*
: The file system in which the file or directory is located. Required.

acl/*path*
: The path of the file or directory for which you want to set the ACL. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 65. List of parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
path	The file path relative to file system's mount point. The path of the file or directory is specified with forward slashes (/). For example: `mnt/gpfs0/rest01`. The forward slashes in the path are encoded to "%2F" in the request URL.	Required.

Table 65. List of parameters (continued)		
Parameter name	**Description and applicable keywords**	**Required/optional**
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
            "type":"{NFSv4} ",
          "entries": [
                 "type":"{allow | deny | alarm | audit }",
                 "who":"User or group",
                 "permissions":"Access permissions",
                 "flags":"Flags",
           ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"type":"*NFSv4*"
Type of the ACL.

"entries":"*Access control entries*"

"type":"allow | deny | alarm | audit"
Type of the entry.

"who":" special:owner@ | special:group@ | special:everyone@ | user:{name} | group:{name}"
The name of the user or group of users for which the ACL is applicable.

"permission":"(r) read | (w) write | (m) mkdir, | (x) execute | (d) delete | (D) delete child | (a) read attr | (A) write attr (n) read named | (N) write Named | (c) read acl | (C) write acl | (o) change owner| (s) synchronize "
The access permissions.

"flags":"(f) file inherit | (d) dir inherit | (i) inherit only | (I) inherited | (S) successful access | (F) failed access"
Special flags and inheritance definition.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
```

```
          }
        "jobId":"ID",
        "submitted":"Time",
        "completed":Time",
        "status":"Job status",
        }
  ],
 }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":"*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example sets ACL information for the file system `gpfs0` and path `mnt/gpfs0`.

Request data:

```
{
  "type": "NFSv4",
  "entries": [
```

```
    {
      "type": "allow",
      "who": "user:testuser",
      "permissions": "rxancs",
      "flags": "fd"
    }
  ]
}
```

Corresponding request URL:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
-d '{
  "type": "NFSv4",
  "entries": [
    {
      "type" : "allow",
      "who" : "special:owner@",
      "permissions" : "rwmxDaAnNcCos",
      "flags" : ""
     },
     {
       "type" : "allow",
       "who" : "special:group@",
       "permissions" : "rxancs",
       "flags" : ""
     },
       {
         "type" : "allow",
         "who" : "special:everyone@",
         "permissions" : "rxancs",
         "flags" : ""
        },
       {
         "type" : "allow",
         "who" : "user:scalemgmt",
         "permissions" : "rxancs",
         "flags" : "fd"
       }
  ]
}' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/acl/mnt%2Fgpfs0'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000002,
    "status" : "RUNNING",
    "submitted" : "2017-03-14 15:50:00,493",
    "completed" : "N/A",
    "request" : {
      "data" : {
        "entries" : [
      {
          "type" : "allow",
          "who" : "special:owner@",
          "permissions" : "rwmxDaAnNcCos",
          "flags" : ""
        },
         {
          "type" : "allow",
          "who" : "special:group@",
          "permissions" : "rxancs",
          "flags" : ""
        },
        {
          "type" : "allow",
          "who" : "special:everyone@",
          "permissions" : "rxancs",
          "flags" : ""
        },
        {
          "type" : "allow",
```

```
          "who" : "user:scalemgmt",
          "permissions" : "rxancs",
          "flags" : "fd"
        } ],
        "type" : "NFSv4"
      },
      "type" : "PUT",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/acl/mnt%2Fgpfs0"
    },
    "result" : { }
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing"
  }
}
```

“mmgetacl command” on page 428
Displays the GPFS access control list of a file or directory.

“mmputacl command” on page 617
Sets the GPFS access control list for the specified file or directory.

Filesystems/{filesystemName}/afm/state: GET

Gets the Active File Management (AFM) state in a file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET filesystems/{filesystemName}/afm/state` request gets AFM state in a file system. For more information about the fields in the data structures that are returned, see the topics “mmafmconfig command” on page 45, “mmafmctl command” on page 61, and “mmafmlocal command” on page 78.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/afm/state
```

where

filesystems/filesetName
Specifies the file system to which the AFM fileset belongs. Required.

afm/state
Specifies AFM as the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request data

No request data.

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 66. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.

Response data

```
{
  "status": {
    "code": 200,
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/filesystems/gpfs0/filesets?lastId=1001"
  },
  "filesetAfmStateList": "null"
  "filesetAfmState" {
          "filesystemName: "File system name"
          "filesetName": "Fileset name"
          "filesetId": "Fileset ID"
          "filesetTarget": "Fileset target"
          "cacheState": "Cache state"
          "gatewayNodeName": Gateway node name"
          "queueLength": "Queue length"
          "drState": "DR state"
          "oid": "OID"
}
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"filesetAfmstateList"

"filesetAfmState":"*AFM state of the fileset*"
Status of AFM filesets that belong to the file system.

"filesystemName": "*File system name*"
Name of the file system to which the AFM fileset belongs.

"filesetName": "*Fileset name*"
Name of the AFM fileset.

"filesetId": "*Fileset ID*"
Unique identifier of the AFM fileset.

"filesetTarget": "*Fileset target*"
The target fileset in the AFM and AFM DR configuration.

"cacheState": "*Cache state*"
The status of the cache site in the AFM configuration.

"gatewayNodeName": "*Gateway node name*"
The name of the gateway node that manages I/O in the AFM configuration.

"queueLength": "*Queue length*"
The length of the AFM queue.

"drState": "*DR state*"
The DR status of the AFM fileset.

"oid": "*OID*"
Internal identifier of the AFM fileset state.

Examples

The following example gets the details of the call home configuration in the system.

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/afm/state'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.


```
{
"filesetAfmStateList" : [ {
"cacheState" : "Active",
"filesetName" : "primary_rpo",
"filesystemName" : "gpfs0"
}, {
"cacheState" : "Active",
"filesetName" : "primary",
"filesystemName" : "gpfs0"
}, {
"cacheState" : "Active",
"filesetName" : "ind_writer_mapped",
"filesystemName" : "gpfs0"
}, {
"cacheState" : "Active",
"filesetName" : "ind_writer_dnsmap",
"filesystemName" : "gpfs0"
}, {
"cacheState" : "Active",
"filesetName" : "ind_writer_cache",
"filesystemName" : "gpfs0"
} ],
"status" : {
"code" : 200,
"message" : "The request finished successfully"
}
}
```


“mmafmlocal command” on page 78
Provides a list of cached files and file statistics such as inode number, allocated blocks, and so on.

“mmafmconfig command” on page 45
Can be used to manage home caching behavior and mapping of gateways and home NFS exported servers.

“mmafmctl command” on page 61
This command is for various operations and reporting information on all filesets. It is recommended to read the *IBM Spectrum Scale: Administration Guide* AFM and AFM Disaster Recovery chapters along with this manual for detailed description of the functions.

Filesystems/{filesystemName}/audit: PUT

Enable or disable file audit logging for a file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `/filesystems/{filesystemName}/audit` request enables or disables file audit logging for a specific file system. For more information about the fields in the data structures that are returned, see “mmaudit command” on page 92.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/filesystemName/
audit
```

where

filesystems/filesystemName/audit
Specifies that file audit logging is going to be enabled or disabled for the specific file system. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 67. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.

Request data

```
{
         "action": "enable | disable"
         "fileAuditLogConfig":
          {

         "auditFilesetDeviceName": "Log File system"
         "auditFilesetName": "Log Fileset",
         "auditRetention": "Retention Period",
         "eventTypes": "Event types",
          }
}
```

"action": "*enable | disable*"
Enable or disable file audit logging for the specified file system.

"FileAuditLogConfig"

"auditFilesetDeviceName": "*logFilesystem* "
The name of the file system where the filesets that contain the audit log records reside.

"auditFilesetName": "logFileset"
The name of the fileset in which the file audit log records must reside.

"auditRetention": "*RetentionPeriod* "
The number of days for which the file audit logs are retained.

"eventTypes": "Event types"
The event types that are included in the file audit logs.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example enables file audit logging for the file system *fs1*.

Request URL:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json' '{ \
   "action": "enable", \
   "fileAuditLogConfig": { \
     "auditFilesetDeviceName": "logFilesystem", \
     "auditFilesetName": "logFileset1", \
     "auditRetention": 365, \
     "eventTypes":
"ACLCHANGE,CLOSE,CREATE,DESTROY,GPFSATTRCHANGE,OPEN,RENAME,RMDIR,UNLINK,XATTRCHANGE" \
   } \
 }'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/fs1/audit'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000004,
    "status" : "COMPLETED",
    "submitted" : "2020-09-29 15:02:11,560",
    "completed" : "2020-09-29 15:02:13,131",
```

```
    "runtime" : 1571,
    "request" : {
      "type" : "PUT",
      "url" : "/scalemgmt/v2/filesystems/fs1/audit"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmaudit 'fs1' enable --log-fileset 'logFileset1'
--log-fileset-device 'logFilesystem' --retention 365 --events
'ACLCHANGE,CLOSE,CREATE,DESTROY,GPFSATTRCHANGE,OPEN,RENAME,RMDIR,UNLINK,XATTRCHANGE' " ],
      "stdout" : [ ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

“mmaudit command” on page 92
Manages setting and viewing the file audit logging configuration in IBM Spectrum Scale.

Filesystems/{filesystemName}/directory/{path}: POST

Creates a directory inside a file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `filesystems/{filesystemName}/directory/{path}` request creates directory inside a file system.

Request URL

```
https://IP address or host name of API server:port/scalemgmt/v2/filesystems/filesystemName/
directory/path
```

where:

`filesystemName`
Specifies the name of the file system to which the directory belongs. Required.

`path`
Specifies the path of the directory. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 68. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system where the directory is going to be created.	Required.
path	The directory path relative to the file system's mount point.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
"user": "User name",
"uid": "User ID",
"group": "Group name",
"gid": "Group ID"
"permission": Permissions
   }
```

"user": "*User name*"
The name of the owning user.

"uid": "*User ID*"
The ID of the owning user.

"group": "*Group name*"
The name of the owning user group.

"gid": "*Group ID*"
The ID of the owning user group.

"permission": *Access permissions*
The number of permissions that are set by using the CLI command **chmod**. If nothing is specified, then no action is allowed.

"recursive": *true | false*
Specifies whether the directory is created recursively within the entire file system or within the specific directory that is defined in the endpoint path. The **recursive** parameter is always displayed as `false` for this request.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":{
               "gid": Group ID,
          "group": "Group name",
          "permissions": Number of permissions,
          "recursive": true | false,
          "uid": User ID,
          "user": "user name"

           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
   ],
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*.

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "

"user":"*User name*"
Name of the owner.

"uid":"*User ID*"
Unique identifier of the owner.

"group":"*Group name*"
Name of the user group that owns the file or directory.

"gid":"*Group ID*"
Unique identifier of the user group that owns the file or directory.

"permissions":"*Number of permissions*"
The number of permissions that are set by using the CLI command **chmod**.

"recursive": *true | false*
Specifies whether the directory is created recursively within the entire file system or only in the specific directory that is defined in the endpoint path. The **recursive** parameter is always displayed as `false` for this request.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to create a directory inside the file system `gpfs0`.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
-d '{
   "user": "testuser55",
   "uid": 1234,
   "group": "mygroup",
   "gid": 4711,
```

```
   "permissions": 777
}' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/directory/mydir'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.


```
{
  "jobs": [
    {
      "jobId": 1000000000001,
      "status": "COMPLETED",
      "submitted": "2021-07-08 09:00:53,498",
      "completed": "2021-07-08 10:00:00,440",
      "runtime": 12,
      "request": {
        "data": {
          "gid": 4711,
          "group": "mygroup",
          "permissions": 700,
          "recursive": false,
          "uid": 1234,
          "user": "testuser55"
        },
        "type": "POST",
        "url": "/scalemgmt/v2/filesystems/gpfs0/directory/mydir"
      },
      "result": {},
      "pids": []
    }
  ],
  "status": {
    "code": 202,
    "message": "The request finished successfully."
  }
}
```

Filesystems/{filesystemName}/directory/{path}: DELETE

Removes a directory from a file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `DELETE filesystems/{filesystemName}/directory/{path}` request removes a directory from a file system.

Request URL

```
https://management API host:port/scalemgmt/v2/filesystems/filesystemName/directory/path
```

where:

filesystems/filesystemName
Specifies the name of the file system to which the directory belongs. Required.

directory/path
The path of the directory to be removed.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 69. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name.	Required.
path	The directory path relative to the file system's mount point.	Required.

Request data

None.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
```

```
            "stdout":"String",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted":"Time",
        "completed":Time",
        "status":"Job status",
        }
 ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to remove a directory from the fileset `gpfs0`.

Request data:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/directory/myDir1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000006,
    "status" : "RUNNING",
    "submitted" : "2017-03-14 16:05:30,960",
    "completed" : "N/A",
    "request" : {
         "type" : "DELETE",
         "url" : "/scalemgmt/v2/filesystems/gpfs0/directory/myDir1"
    },
    "result" : { }
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing"
  }
}
```

Filesystems/{filesystemName}/directoryCopy/{sourcePath}: PUT

Copies a directory on a GPFS file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `filesystems/{filesystemName}/directoryCopy/{sourcePath}` request copies a directory on a particular file system. For more information about the fields in the data structures that are returned, see the topics "mmcrfs command" on page 318, "mmchfs command" on page 232, and "mmlsfs command" on page 506.

Request URL

```
https://<IP or host name of API server>:port/scalemgmt/v2/filesystems/{filesystemName}/
directoryCopy/{sourcePath}
```

where:

filesystems/filesystemName
Specifies the name of the file system to which the directory belongs. Required.

directoryCopy
Action to be performed on the directory. Required.

sourcePath
Path of the directory to be copied. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 70. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
sourcePath	Path of the directory to be copied.	Required.

Request data

```
{
"targetFilesystem": "File system name",
"targetFileset": "Fileset name",
"targetPath": "Directory path",
"nodeclassName": "Name of the node class",
"force": "True | False",
}
```

"targetFilesystem": "*File system name*"
The name of the file system where the directory is located.

"targetFileset": "*Fileset name*"
The name of the fileset where the directory is located. This is optional.

"targetPath": "*Directory path*"
The name of the file system where the directory is located.

"nodeclassName": "*Name of the node class*"
The name of the node class.

"force": "*File system name*"
Specifies whether the **cp** command is used with **--force** option.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to copy a directory that belongs to the file system `gpfs0`.

Request data:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
-d '{
  "targetFilesystem": "gpfs0", \
   "targetFileset": "fset1", \
   "targetPath": "mydir", \
   "nodeclassName": "cesNodes", \
   "force": true \
}' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/directoryCopy/mydir'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000009,
    "status" : "COMPLETED",
    "submitted" : "2020-09-29 15:39:55,452",
    "completed" : "2020-09-29 15:39:58,675",
    "runtime" : 3223,
    "request" : {
      "type" : "PUT",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/directoryCopy/mydir"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "tscp --source '/mnt/gpfs0/mydir' --target '/mnt/gpfs0/fset1/mydir' --
nodeclass 'cesNodes' --force " ],
      "stdout" : [ ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

See also

- “mmcrfs command” on page 318
- “mmchfs command” on page 232
- “mmlsfs command” on page 506

Filesystems/{filesystemName}/disks: GET

Gets details of the disks that are part of a file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET filesystems/{filesystemName}/disks` request gets details of the disks that belong to the specified file system. For more information about the fields in the data structures that are returned, see the topics "mmlsdisk command" on page 497, "mmlsnsd command" on page 522and "mmlsfs command" on page 506.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/filesystemName/
disks
```

where

filesystems/filesystemName
Specifies the file system to which the disks belong. Required.

disks
The disks that belong to the specified file system. Required.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 71. List of parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   },
 "paging":
  {
    "next": "URL"
  },
   disks: [
      {
         "name":"DiskName",
         "filesystem":"FileSystemName",
         "failureGroup":"FailureGroupID",
         "type":"{dataOnly | metadataOnly | dataAndMetadata | descOnly | localCache}",
         "storagePool":"StoragePool",
         "status":"Status",
         "availability":Availability,
         "quorumdisk":"QuorumDisk",
         "remarks":Remarks,
         "size":Size,
         "availableBlocks":"AvailableBlocks",
         "availableFragments":"AvailableFragments"
         "nsdServers":"NSDServers",
         "nsdVolumeId":"NSDVolumeID",
       }
   ],

}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"disks":
An array of elements that describe one disk.

"name":"*DiskName*"
Name of the disk.

"filesystem":"*FileSystemName*"
The file system to which the disk belongs.

"failureGroup":"*FailureGroupID*"
Failure group of the disk.

"type":"*{dataOnly | metadataOnly | dataAndMetadata | descOnly | localCache}*"
Specifies the type of data to be stored on the disk:

dataOnly
Indicates that the disk contains data and does not contain metadata. This is the default for disks in storage pools other than the system pool.

metadataOnly
Indicates that the disk contains metadata and does not contain data.

dataAndMetadata
Indicates that the disk contains both data and metadata. This is the default for disks in the system pool.

descOnly
Indicates that the disk contains no data and no file metadata. Such a disk is used solely to keep a copy of the file system descriptor, and can be used as a third failure group in certain disaster recovery configurations. For more information, see the help topic *Synchronous mirroring utilizing GPFS replication* in *IBM Spectrum Scale: Administration Guide*.

localCache
Indicates that the disk is to be used as a local read-only cache device.

"storagePool":"*StoragePool*"
Storage pool to which the disk belongs.

"status":"*Status*"
The status of the disk.

"availability":"*Availability*"
The availability of the disk.

"quorumdisk":"*Quorum disk*"
The quorum status of the disk.

"remarks":"*Remarks*"
User-defined remarks about the disk.

"size":"*Size*"
The size of the disk.

"availableBlocks":"*Available blocks*"
The available blocks of the disk.

"availableFragments":"*Available fragments*"
The available fragments of the disk.

"nsdServers":"*NSD servers*"
The name of the NSD servers.

"nsdVolumeID":"*NSD volume ID*"
The volume ID of the NSD.

Examples

The following example gets information about disks in the file system `gpfs0`.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/disks'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

The **disks** array returns details of the disks. Each **config** object contains details about one disk.

```
{
  "disks" : [ {
        "fileSystem" : "gpfs0",
      "name" : "disk1"
  }, {
      "fileSystem" : "gpfs0",
      "name" : "disk8"
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully"
  }
}
```

Using the field parameter ":all:" returns entire details of the disks that are available in the system.

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/disks?fields=:all:'
```

```
{
  "disks" : [ {
      "availability" : "up",
      "availableBlocks" : "9.15 GB",
      "availableFragments" : "552 kB",
      "failureGroup" : "1",
      "fileSystem" : "gpfs0",
      "name" : "disk1",
      "nsdServers" : "mari-11.localnet.com,mari-15.localnet.com,mari-14.localnet.com,
                      mari-13.localnet.com,mari-12.localnet.com",
      "nsdVolumeId" : "0A00640B58B82A8C",
      "quorumDisk" : "no",
      "remarks" : "desc",
      "size" : " 10.00GiB",
      "status" : "ready",
      "storagePool" : "system",
      "type" : "nsd"
  }, {
      "availability" : "up",
      "availableBlocks" : "9.45 GB",
      "availableFragments" : "664 kB",
      "failureGroup" : "1",
      "fileSystem" : "gpfs0",
      "name" : "disk8",
      "nsdServers" : "mari-11.localnet.com,mari-12.localnet.com,mari-13.localnet.com,
                      mari-14.localnet.com,mari-15.localnet.com",
      "nsdVolumeId" : "0A00640B58B82AD9",
      "quorumDisk" : "no",
      "remarks" : "desc",
      "size" : " 10.00GiB",
      "status" : "ready",
      "storagePool" : "data",
      "type" : "nsd"
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully"
  }
}
```

“mmlsdisk command” on page 497
Displays the current configuration and state of the disks in a file system.

“mmcrnsd command” on page 335
Creates Network Shared Disks (NSDs) used by GPFS.

“mmcrfs command” on page 318
Creates a GPFS file system.

“mmadddisk command” on page 28
Adds disks to a GPFS file system.

Filesystems/{filesystemName}/disks/{diskName}: GET

Gets details of a specific disk that is part of a file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET filesystems/{filesystemName}/disks/{diskName}` request gets details of a specific disk. For more information about the fields in the data structures that are returned, see the topics “mmlsdisk command” on page 497, “mmlsnsd command” on page 522and “mmlsfs command” on page 506.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/fileSystemName/
disks/diskName
```

where

filesystems/filesystemName
The file system to which the disks belong. Required.

disks/diskName
The disk about which you want to get information. Required.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 72. List of parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
diskName	The name of the disk.	Required.

Request data

No request data.

Response data

```
{
    "status": {
```

```
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
 "paging":
  {
    "next": "URL"
  },
   disks: [
      {
         "name":"DiskName",
         "filesystem":"FileSystemName",
         "failureGroup":"FailureGroupID",
         "type":"{dataOnly | metadataOnly | dataAndMetadata | descOnly | localCache}",
         "storagePool":"StoragePool",
         "status":"Status",
         "availability":"Availability",
         "quorumdisk":"QuorumDisk",
         "remarks":"Remarks",
         "size":"Size",
         "availableBlocks":"AvailableBlocks",
         "availableFragments":"AvailableFragments"
         "nsdServers":"NSDServers",
         "nsdVolumeId":"NSDVolumeID",
       }
   ],

}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"disks":
An array of elements that describe one disk.

"name":"*DiskName*"
Name of the disk.

"filesystem":"*FileSystemName*"
The file system to which the disk belongs.

"failureGroup":"*FailureGroupID*"
Failure group of the disk.

"type":"*{dataOnly | metadataOnly | dataAndMetadata | descOnly | localCache}*"
Specifies the type of data to be stored on the disk:

dataOnly
Indicates that the disk contains data and does not contain metadata. This is the default for disks in storage pools other than the system pool.

metadataOnly
Indicates that the disk contains metadata and does not contain data.

dataAndMetadata
Indicates that the disk contains both data and metadata. This is the default for disks in the system pool.

descOnly
Indicates that the disk contains no data and no file metadata. Such a disk is used solely to keep a copy of the file system descriptor, and can be used as a third failure group in

certain disaster recovery configurations. For more information, see the help topic *Synchronous mirroring utilizing GPFS replication* in *IBM Spectrum Scale: Administration Guide*.

localCache
Indicates that the disk is to be used as a local read-only cache device.

"storagePool":"*StoragePool*"
Storage pool to which the disk belongs.

"status":"*Status*"
The status of the disk.

"availability":"*Availability*"
The availability of the disk.

"quorumdisk":"*Quorum disk*"
The quorum status of the disk.

"remarks":"*Remarks*"
User-defined remarks about the disk.

"size":"*Size*"
The size of the disk.

"availableBlocks":"*Available blocks*"
The available blocks of the disk.

"availableFragments":"*Available fragments*"
The available fragments of the disk.

"nsdServers":"*NSD servers*"
The name of the NSD servers.

"nsdVolumeID":"*NSD volume ID*"
The volume ID of the NSD.

Examples

The following example gets information about the disk `disk8` in the file system `gpfs0`.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/disks/disk8'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "paging": {
    "next": "https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/disks/disk8?lastId=1001"
  },
  "disks": [
    {
        "name": "disk8",
        "fileSystem": "gpfs0",
        "failureGroup": "1",
        "type": "dataOnly",
        "storagePool": "data",
        "status": "ready",
        "availability": "up",
        "quorumDisk": "no",
        "remarks": "This is a comment",
        "size": "10.00 GB",
        "availableBlocks": "730.50 MB",
        "availableFragments": "1.50 MB",
```

```
        "nsdServers": "gpfsgui-21.localnet.com",
        "nsdVolumeId": "0A0064155874F5AA"
    }
  ]
}
```

“mmlsdisk command” on page 497
Displays the current configuration and state of the disks in a file system.

“mmcrnsd command” on page 335
Creates Network Shared Disks (NSDs) used by GPFS.

“mmcrfs command” on page 318
Creates a GPFS file system.

“mmadddisk command” on page 28
Adds disks to a GPFS file system.

Filesystems/{filesystemName}/filesets: GET

Gets information of all filesets that are part of a specific file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET filesystems/{filesystemName}/filesets` request gets information about all filesets in the specified file system. For more information about the fields in the data structures that are returned, see the topics “mmcrfileset command” on page 311, “mmchfileset command” on page 224, and “mmlsfileset command” on page 501.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filsystems/filesystemName/
filesets
```

where:

`filesystem/filesystemName`
Specifies the name of the file system to which the filesets belong. Required.

`filesets`
Specifies that you need to get details of all filesets that belong to the specified file system. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 73. List of parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Request data

No request data.

Response data

```
{
  "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   },
 "paging":
  {
    "next": "URL",
    "fields": "Fields",
    "filter": "Filters",
    "baseUrl": "Base URL",
    "lastId": "ID of the last element",
  },
   "filesets": [
      {
            "filesetName":"Fileset",
            "filesystemName":"Device",
            "config":
             {
               "path": "Path",
               "inodeSpace":"Inodes",
               "maxNumInodes":"Inodes",
               "permissionChangeMode":"Mode",
               "comment":"Comment",
               "iamMode":"Mode",
               "oid": "ID",
               "id":"ID",
               "status":"Status"
               "parentId":"ID",
               "created":"DateTime",
               "isInodeSpaceOwner":"isOwner",
               "inodeSpaceMask":"inodeSpace",
               "snapID":"ID",
               "rootInode":"Inode",
          }
        "afm":
          {
            "afmTarget":"Protocol://{Host | Map}/Path",
            "afmState":"enabled | disabled",
            "afmMode": "{single-writer | sw | read-only | ro | local-updates | lu |
               independent-writer | Primary | Secondary}",
            "afmFileLookupRefreshInterval":"Interval",
            "afmDirLookupRefreshInterval":"Interval",
            "afmDirOpenRefreshInterval":"Interval",
            "afmAsyncDelay":"Delay",
            "afmNeedsRecovery":"true | false"
            "afmExpirationTimeout":"Interval",
            "afmRPO":"Interval",
            "afmLastPSnapId":"Psnap ID"
            "afmShowHomeSnapshots":"{yes | no}",
            "afmNumReadThreads":"Number of read threads",
            "afmReadBufferSize":"Read buffer size",
            "afmWriteBufferSize":"Write buffer size",
            "afmReadSparseThreshold":"Read sparse thread",
            "afmParallelReadChunkSize":"Size",
            "afmParallelReadThreshold":"Threshold",
            "afmNumFlushThreads":"Threads",
            "afmPrefetchThreshold":"{1 | 1-99 | 100}",
            "afmEnableAutoEviction":"{yes | no}",
            "afmParallelWriteThreshold":"Threshold",
            "afmNeedsResync":"true | false",
            "afmParallelWriteChunkSize":"Size",
            "afmNumWriteThreads":"Number of write threads",
            "afmPrimaryID":"ID",
            "afmDRState":"DR state"
            "afmAssociatedPrimaryId":"ID"
            "afmDIO":"",
            "afmHostMapName":"Name",
            "afmHostMapping":"Mapping details"[
              {
               "home":"IP or host name of home",
               "cache":"IP or hostname of cache",
              }
             ]
           "afmGatewayNode":"IP or host name of home",
           "afmIOFlags":"I/O flag",
           "afmVerifyDmapi":"true | false",
           "afmSkipHomeACL":"true | false",
           "afmSkipHomeMtimeNsec":"true | false",
```

```
            "afmForceCtimeChange":"true | false",
            "afmSkipResyncRecovery":"true | false",
            "afmSkipConflictQDrop":"true | false",
            "afmRefreshAsync":"true | false",
            "afmParallelMounts":"true | false",
            "afmRefreshOnce":"true | false",
            "afmSkipHomeCtimeNsec":"true | false",
            "afmReaddirOnce":"true | false",
            "afmResyncVer2":"true | false",
            "afmSnapUncachedRead":"true | false",
            "afmFastCreate":"true | false",
            "afmObjectXattr":"true | false",
            "afmObjectVHB":"true | false",
            "afmObjectBlkIO":"true | false",
            "afmSkipHomeRefresh":"true | false",
            "afmObjectGCS":"true | false",
            "afmObjectUserKeys":"true | false",
            "afmWriteOnClose":"true | false",
            "afmObjectSubdir":"true | false",
            "afmObjectSSL":"true | false",
            "cacheState":"State of the cache",
            "queueLength":"Queue length",
            "queueNumExec":"Queue number",
            "afmHostMaps":"Host map details",[
              {
              "home":"IP address or hostname of the home",
              "cache":"IP address or hostname of the cache",
              }
              ]
        },
    "usage":
      {
       "usedInodes":"Inodes used",
       "allocatedInodes":"Inodes allocated",
       "usedBytes":"Used capacity",
       "inodeSpaceUsedInodes":"Inode space used",
       "inodeSpaceFreeInodes":"Inode space available",
      }
}
}
```

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code":"*ReturnCode*"
The return code.

"paging":
Paging details.

"next": "*URL*",
The URL to retrieve the next page. Paging is enabled when more than 1000 objects would be returned by the query.

"fields": "*Fields*",
The fields used in the original request.

"filter": "*Filters*",
The filter used in the original request.

"baseUrl": "*Base URL*",
The URL of the request without any parameters.

"lastId": "*ID of the last element*",
The ID of the last element that can be used to retrieve the next elements.

"*filesets*":
An array of information about the filesets in the specified file system. Each array element describes one fileset and can contain the data structures `afm`, `config`, `links`, and `state`. For more information about the fields in these data structures, see the links at the end of this topic:

"filesets":
Information about the fileset configuration.

"filesetName": "*Fileset*",
The name of the fileset.

"filesystemName": "*Device*"
Required. The file system in which the fileset is located.

"config":

"path": "*Path*"
The absolute path of the fileset.

"inodeSpace": "*Inodes*"
The number of inodes that are allocated for use by the fileset.

"maxNumInodes": "*Inodes*"
The inode limit for the inode space that is owned by the specified fileset.

"permissionChangeMode": "*Mode*"
The permission change mode. Controls how chmod and ACL commands affect objects in the fileset.

chmodOnly
Only the chmod command can change access permissions.

setAclOnly
Only the ACL commands and API can change access permissions.

chmodAndSetAcl
Both the chmod command and ACL commands can change access permissions.

chmodAndUpdateAcl
Both the chmod command and ACL commands can change access permissions.

"comment": "*Comment*",
A comment that appears in the output of the `mmlsfileset` command.

"iamMode": "*Mode*"
The integrated archive manager (IAM) mode for the fileset.

ad | advisory

nc | noncompliant

co | compliant

"oid": "*ID*"
Internal identifier of the fileset that is used for paging.

"id": *ID*
The fileset identifier.

"status": "*Status*"
Specifies whether the file is linked or unlinked.

"parentId": *ParentID*
The parent identifier of the fileset.

"created": "*DateTime*",
The date and time when the fileset was created.

"isInodeSpaceOwner": "*isOwner*",
Indicates whether the fileset has its own inode space.

"isInodeSpaceMask": "*isMask*",
The inode space mask.

"snapID": *ID*
The snapshot ID.

"afm":
Information about Active File Management (AFM).

"afmTarget": "*Interval*"*Protocol*://{Host | Map}/Path"
The home that is associated with the cache.

"afmState":"*enabled | disabled*"

"afmMode": "{single-writer | sw | read-only | ro | local-updates | lu | independent-writer | Primary | Secondary}"
The mode in which the cache operates.

"afmFileLookupRefreshInterval": *Interval*
The interval in seconds between data revalidations of files that are caused by lookup operations such as `ls` and `stat`.

"afmDirLookupRefreshInterval":" *Interval*"
The interval in seconds between data revalidations of directories that are caused by lookup operations such as `ls` and `stat`.

"afmDirOpenRefreshInterval": *Interval*
The interval in seconds between data revalidations of directories that are caused by lookup operations.

"afmAsyncDelay": "*Delay*"
The time in seconds by which to delay write operations because of the lag in updating remote clusters.

"afmNeedsRecovery":"*true | false*"

"afmExpirationTimeout": *Interval*
The timeout in seconds after which cached data is considered expired. Used with `afmDisconnectTimeout`, which is set with `mmchconfig`.

"afmRPO": *Interval*
The recovery point objective (RPO) interval, in minutes, for a primary fileset.

"afmLastPSnapId":"*Psnap ID*"

"afmShowHomeSnapshots": "{yes | no}"
Shows or hides the home snapshot directory in cache. Specify `yes` to show, `no` to hide.

"afmNumReadThreads":"*Number of read threads*"
Number of AFM read threads

"afmReadBufferSize":"*Read buffer size*"
AFM read buffer size.

"afmWriteBufferSize":"*Write buffer size*"
AFM write buffer size.

"afmReadSparseThreshold":"*Read sparse thread*"
Read sparse threshold.

"afmParallelReadChunkSize": *Size*
The minimum chunk size of read data, in bytes, that must be distributed among the gateway nodes during parallel reads.

"afmParallelReadThreshold": *Threshold*
The minimum file size, in megabytes, at which to do parallel reads.

"afmNumFlushThreads": *numThreads*
The number of threads that are used on each gateway to synchronize updates to the home cluster.

"afmPrefetchThreshold": {1 | 1-99 | 100}
The percentage of file size that must be cached before the entire file is prefetched.

0
Enables full file prefetching.

1-99
The percentage of file size that must be cached before the entire file is prefetched.

100
Disables full file prefetching.

"afmEnableAutoEviction": "{yes | no}"
Enables or disables eviction on the fileset.

"afmParallelWriteThreshold": ***Threshold***
The minimum file size, in megabytes, at which to do parallel writes.

"afmNeedsResync":"*true | false*"
Whether the AFM needs resynchronization.

"afmParallelWriteChunkSize": ***Size***
The minimum chunk size of write data, in bytes, that must be distributed among the gateway nodes during parallel writes.

"afmNumWriteThreads":"*Number of write threads*"

"afmPrimaryID": "*ID*"
The unique primary ID of the primary fileset for asynchronous data replication.

afmDRState":"*DR state*"
Status of the AFM DR.

"afmAssociatedPrimaryId":"*ID*"
Associated primary ID.

"afmDIO":"*Number*"
IO details.

"afmHostMapName":"*Name*"
The name of the AFM host map this fileset belongs to

"afmHostMapping":

"home":"*Home address*"
IP or hostname of home.

"cache":"*Cache address*"
IP or hostname of cache

"afmGatewayNode":"*Gateway node*",
Name of the gateway node.

"afmIOFlags":"*I/O flag*"
AFM I/O flag.

"afmVerifyDmapi":"*true | false*"
Whether to verify AFM DMAPI.

"afmSkipHomeACL":"*true | false*"
Whether to ignore home ACL.

"afmSkipHomeMtimeNsec":"*true | false*"
Whether to skip home *mtime* and *nsec*.

"afmForceCtimeChange":"*true | false*"
Whether to force change *ctime*.

"afmSkipResyncRecovery":"*true | false*"

"afmSkipConflictQDrop":"*true | false*"

"afmRefreshAsync":"*true | false*"

"afmParallelMounts":"*true | false*"

"afmRefreshOnce":"*true | false*"

"afmSkipHomeCtimeNsec":"*true | false*"

"afmReaddirOnce":"*true | false*"

"afmResyncVer2":"*true | false*"

"afmSnapUncachedRead":"*true | false*"

"afmFastCreate":"*true | false*"

"afmObjectXattr":"*true | false*"
Whether user extended attributes are available in object store.

"afmObjectVHB":"*true | false*"
Whether to use Virtual Hosted Bucket, for Alibaba Clod Object Storage Service (OSS).

"afmObjectBlkIO":"*true | false*"

"afmSkipHomeRefresh":"*true | false*"

"afmObjectGCS":"*true | false*"
Whether to use Google Cloud Services.

"afmObjectUserKeys":"*true | false*"
Whether to use user keys on requests from the object store.

"afmWriteOnClose":"*true | false*"

"afmObjectSubdir":"*true | false*"
Whether to create sub-directory if '/' is present in the object name.

"afmObjectSSL":"*true | false*"
Whether to enable SSL certificate verification, which is valid only with HTTPS from the object store.

"cacheState":"*State of the cache*"
Current cache state.

"queueLength":""
Length of the queue on the primary gateway.

"queueNumExec":""
Number of operations played at home since the fileset is last active.

"afmHostMaps":""

"home":""
IP address or hostname of the home site.

"cache":""
IP address or hostname of the cache site.

"usage":""

"usedInodes":"*Used inodes*"
Specifies the used inodes for the fileset.

"allocatedInodes":"*allocated inodes*"
Specifies the allocated inodes for the fileset.

"usedBytes":"*Used bytes*"
Specifies the used bytes for the fileset.

"inodeSpaceUsedInodes":"*Used inode space*"
Specifies the used inode of the fileset.

"inodeSpaceFreeInodes":"*Free inodes*"
Specifies the number of inode that are available for the fileset.

Examples

The following example gets information about the filesets in file system `gpfs0`.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "filesets" : [ {
    "config" : {
      "filesetName" : "root",
      "filesystemName" : "gpfs0"
    }
  }, {
    "config" : {
      "filesetName" : "fset1",
      "filesystemName" : "gpfs0"
    }
  }, {
    "config" : {
      "filesetName" : "fset2",
      "filesystemName" : "gpfs0"
    }
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully"
  }
}
```

Using the field parameter ":all:" returns entire details of the filesets that are part of the specified file system. For example:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets?fields=:all:'
```

```
{
  "status": {
    "code": 200,
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/filesystems/gpfs0/filesets?lastId=10001",
    "fields": "period,restrict,sensorName",
    "filter": "usedInodes>100,maxInodes>1024",
    "baseUrl": "/scalemgmt/v2/perfmon/sensor/config",
    "lastId": 10001
  },
  "filesets": [
    {
      "filesetName": "myFset1",
      "filesystemName": "gpfs0",
      "config": {
        "path": "/mnt/gpfs0/myFset1",
        "inodeSpace": 0,
        "maxNumInodes": 4096,
        "permissionChangeMode": "chmodAndSetAcl",
        "comment": "Comment1",
        "iamMode": "off",
        "oid": 158,
        "id": 5,
        "status": "Linked",
        "parentId": 1,
        "created": "2016-12-13 13.59.15",
        "isInodeSpaceOwner": false,
        "inodeSpaceMask": 0,
        "snapId": 0,
        "rootInode": 131075
      },
      "afm": {
        "afmTarget": "nfs://10.0.100.11/gpfs/afmHomeFs/afmHomeFileset2",
        "afmState": "enabled",
        "afmMode": "read-only",
        "afmFileLookupRefreshInterval": 30,
        "afmFileOpenRefreshInterval": 0,
        "afmDirLookupRefreshInterval": 60,
        "afmDirOpenRefreshInterval": 60,
        "afmAsyncDelay": 0,
        "afmNeedsRecovery": true,
        "afmExpirationTimeout": 100,
        "afmRPO": 0,
```

```
        "afmLastPSnapId": 0,
        "afmShowHomeSnapshots": false,
        "afmNumReadThreads": 0,
        "afmReadBufferSize": 0,
        "afmWriteBufferSize": 0,
        "afmReadSparseThreshold": 0,
        "afmParallelReadChunkSize": 134217728,
        "afmParallelReadThreshold": 1024,
        "afmNumFlushThreads": 4,
        "afmPrefetchThreshold": 0,
        "afmEnableAutoEviction": false,
        "afmParallelWriteThreshold": 0,
        "afmNeedsResync": true,
        "afmParallelWriteChunkSize": 0,
        "afmNumWriteThreads": 4,
        "afmPrimaryID": "string",
        "afmDRState": "string",
        "afmAssociatedPrimaryId": "string",
        "afmDIO": 0,
        "afmHostMapName": "map1",
        "afmHostMapping": [
          {
            "home": "10.0.100.13",
            "cache": "cache-41"
          }
        ],
        "afmGatewayNode": "testnode-11",
        "afmIOFlags": "0x4000000",
        "afmVerifyDmapi": true,
        "afmSkipHomeACL": true,
        "afmSkipHomeMtimeNsec": true,
        "afmForceCtimeChange": true,
        "afmSkipResyncRecovery": true,
        "afmSkipConflictQDrop": true,
        "afmRefreshAsync": true,
        "afmParallelMounts": true,
        "afmRefreshOnce": true,
        "afmSkipHomeCtimeNsec": true,
        "afmReaddirOnce": true,
        "afmResyncVer2": true,
        "afmSnapUncachedRead": true,
        "afmFastCreate": true,
        "afmObjectXattr": false,
        "afmObjectVHB": false,
        "afmObjectBlkIO": true,
        "afmSkipHomeRefresh": true,
        "afmObjectGCS": false,
        "afmObjectUserKeys": false,
        "afmWriteOnClose": true,
        "afmObjectSubdir": true,
        "afmObjectSSL": false,
        "cacheState": "Active",
        "queueLength": 5,
        "queueNumExec": 10,
        "afmHostMaps": [
          {
            "home": "10.0.100.13",
            "cache": "cache-41"
          }
        ]
      },
      "usage": {
        "usedInodes": 1000,
        "allocatedInodes": 65856,
        "usedBytes": 134217728,
        "inodeSpaceUsedInodes": 1100,
        "inodeSpaceFreeInodes": 55856
      }
    }
  ]
}
```


“mmcrfileset command” on page 311
Creates a GPFS fileset.

“mmchfileset command” on page 224
Changes the attributes of a GPFS fileset.

“mmlsfileset command” on page 501
Displays attributes and status for GPFS filesets.

Nodes/health/config/webhook/deleteEventWebhook: DELETE

Deletes an existing webhook event URL.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `DELETE nodes/health/config/webhook/deleteEventWebhook` request deletes an existing webhook event URL. For more information about the fields in the data structures that are returned, see "mmhealth command" on page 441.

Request URL

```
https://<IP address or host name of API server>:<port>scalemgmt/v2/nodes/health/config/webhook/
deleteEventWebhook
```

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request data

```
{
  "webhookurl": "string"
}
```

The details of the parameters are provided in the following list.

"webhookurl: "Event URL"
The webhook event URL that must be deleted.

Response data

```
{
  "jobs": [
    {
      "jobId": "Job ID",
      "status": "RUNNING | COMPLETED | FAILED",
      "submitted": "Date and Time",
      "completed": "Date and Time",
      "runtime": "Duration",
      "request": {
        "type": "POST | GET | PUT | DELETE",
        "url": "URL"
      },
      "result": {
         "progress" [],
         " commands" [],
         " stdout" [] ,
       },
       "pids": [],
    }
  ],
  "status": {
    "code": "Return Code",
    "message": "Return Message."
  }
}
```

For more information about the fields in the following data structures, see the link at the end of the topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"jobId": "*ID*",
The unique ID of the job.

"status": *RUNNING | COMPLETED | FAILED*
Return status.

"submitted": "*Time*"
The date and time at which the job was submitted.

"completed": "*Time*"
The date and time at which the job was completed.

"runtime": "*Time*"
The duration for which the job ran.

"request"
The request details.

"type": "*GET | POST | PUT | DELETE* "
The HTTP request type.

URL
The URL through which the job was submitted.

result
The request result.

"progress": "*string*"
Progress information for the request.

"commands": "*Array*"
The array of commands that are run in this job.

"stdout": "*string*"
CLI messages from stdout.

"pids": "Process ID"
The process IDs of all the active sub processes that manage the job.

status
The status of the request message.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

Examples

The following API command deletes the webhook event URL `http://www.ibm.com`.

Request URL:

```
curl -X DELETE --header 'Content-Type: application/json' --header 'Accept: application/json'
--header 'Authorization: Basic YWRtaW46YWRtaW4wMDE=' -d '{ \ "webhookurl": "http://www.ibm.com"
\ }' '
'https://198.51.100.1:443/scalemgmt/v2/nodes/health/config/webhook/deleteEventWebhook'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
    "jobs": [
        {
            "jobId": 1000000000003,
            "status": "COMPLETED",
            "submitted": "2021-05-03 01:55:41,033",
            "completed": "2021-05-03 01:55:41,450",
            "runtime": 417,
            "request": {
                "type": "DELETE",
                "url": "/scalemgmt/v2/nodes/health/config/webhook/deleteEventWebhook"
            },
            "result": {
                "progress": [],
                "commands": [
                    "mmhealth config webhook remove 'http://www.ibm.com' "
                ],
                "stdout": [
                    "Webhook URL http://www.ibm.com successfully removed from health event
monitoring",
                    "info: Webhook URL http://www.ibm.com successfully removed from health
event monitoring\n"
                ],
```

Related reference

“mmhealth command” on page 441
Monitors health status of nodes.

Nodes/health/config/webhook/listEventWebhook: GET

Gets a list of webhook event URLs.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET nodes/health/config/webhook/listEventWebhook` request gets a list of webhook event URLs. For more information about the fields in the returned data structure, see “mmhealth command” on page 441.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nodes/health/config/webhook/
listEventWebhook
```

where

webhook/listEventWebhook
: Specifies the webhook event URLs as the resource. Required.

Request headers

```
Accept: application/json
```

Parameters

No parameters.

Request data

No request data.

Response data

```
{
  "status": {
    "code": Return Code,
    "message": "Return message."
  },
  "webhookEventList": []

  }
```

The details of the parameters are provided in the following list:

status
: The request status

  "code" : *The Return code*
  : The HTTP Status code.

  "message" : "*Return message*"
  : The return message.

"webhookEventList": *Array*
: The list of overbook events.

Examples

The following example lists the webhook event URLs.

Request data:

```
curl -k -u user:password -X GET --header 'accept:application/json' --header 'Authorization:
Basic YWRtaW46YWRtaW4wMDE=' 'https://198.51.100.1:443/scalemgmt/v2/nodes/health/config/webhook/
listEventWebhook
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": 200,
    "message": "The request finished successfully."
  },
  "webhookEventList": [
    "mmhealth:webhook:HEADER:version:reserved:reserved:url:uuid:status:",
    "mmhealth:webhook:0:1:::http%3A//www.ibm.com:d7044381-2a54-4619-b84f-4b40734d62d5:enabled:"
  ]
}
```

Related reference

“mmhealth command” on page 441
Monitors health status of nodes.

Nodes/health/config/webhook/addEventWebhook: POST

Adds a webhook event URL for monitoring node health status.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `nodes/health/config/webhook/addEventWebhook` request adds a webhook event URL for health event monitoring. For more information about the fields in the data structures that are returned, see “mmhealth command” on page 441.

Request URL

```
https://<IP address or host name of API server>:port/scalemgmt/v2/nodes/health/config/webhook/
addEventWebhook
```

where:

webhook
Specifies the webhook event as the resource. Required.

Request headers

```
Accept: application/json
```

Request data

```
{
  "webhookurl": "URL"
}
```

The details of the parameters are provided in the following list.

"webhookurl: "Event URL"
The webhook event URL that is to be added.

Response data

```
{
  "jobs": [
    {
      "jobId": "Job ID",
      "status": "RUNNING | COMPLETED | FAILED",
      "submitted": "Time",
      "completed": "Time",
      "runtime": "Time",
      "request": {
        "type": "POST | GET | PUT | DELETE",
        "url": "URL"
      },
      "result": {
         "progress" [],
         " commands" [],
         " stdout" [] ,
       },
       "pids": [],
    }
  ],
  "status": {
    "code": Return Code,
    "message": "Return Message."
```

```
  }
}
```

For more information about the fields in the following data structures, see the link at the end of the topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"jobId": "*ID*",
The unique ID of the job.

"status": *RUNNING | COMPLETED | FAILED*
Return status.

"submitted": "*Time*"
The date and time at which the job was submitted.

"completed": "*Time*"
The date and time at which the job was completed.

"runtime" : "*Duration*"
The duration for which the job ran.

"request"
The request details.

"type" : "*GET | POST | PUT | DELETE* "
The HTTP request type.

URL
The request URL.

"result"
The request result.

"progress": "*string*"
Progress information for the request.

"commands": "*string*"
Array of commands that are run in this job.

"stdout" : "*string*"
CLI messages from stdout.

"pids": "Process ID"
The process IDs of all the active sub processes that manage the job.

status
The status of the request message.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example shows how to add a webhook event URL.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json' -d '{ \
   "webhookurl": "http://www.ibm.com" \
}' 'https://198.51.100.1:443/scalemgmt/v2/nodes/health/config/webhook/addEventWebhook'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.


```
{
    "jobs": [
        {
            "jobId": 1000000000001,
            "status": "COMPLETED",
            "submitted": "2021-05-03 01:51:23,674",
            "completed": "2021-05-03 01:51:24,853",
            "runtime": 1179,
            "request": {
                "type": "POST",
                "url": "/scalemgmt/v2/nodes/health/config/webhook/addEventWebhook"
            },
            "result": {
                "progress": [],
                "commands": [
                    "mmhealth config webhook add 'http://www.ibm.com' "
                ],
                "stdout": [
                    "Successfully connected to http://www.ibm.com",
                    "Webhook URL http://www.ibm.com successfully linked to the health event
monitoring system.",
                    "Webhook UUID is 6d863ddb-8a3a-4d4f-adab-225a37e8b4a3",
                    "info: Successfully connected to http://www.ibm.com\nWebhook URL http://
www.ibm.com successfully linked to the health event monitoring system.\nWebhook UUID is
6d863ddb-8a3a-4d4f-adab-225a37e8b4a3\n"
                ],
```


Related reference

“mmhealth command” on page 441
Monitors health status of nodes.

Filesystems/{filesystemName}/filesets: POST

Creates a fileset.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `filesystems/{filesystemName}/filesets` command creates a new fileset in the specified file system. For more information about the fields in the request data structures, see the topics "mmcrfileset command" on page 311, "mmchfileset command" on page 224, and "mmcrfs command" on page 318.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/
{filesystemName}/filesets
```

where

`filesystem/filessystemName`
Specifies the file system in which the fileset is to be created. Required.

`filesets`
Specifies fileset as the resource. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 74. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
      "filesetName": "Fileset name",
      "path": "Path",
      "owner": "UserID[:GroupID]",
      "permissions": "Permissions"
      "inodeSpace": "{new | ExistingFileset}",
      "maxNumInodes": "Inodes",
      "allocInodes": "Inodes",
```

```
      "permissionChangeMode": "chmodOnly | setAclOnly | chmodAndSetAcl",
      "comment": "Comment",
      "iamMode": "{advisory | noncompliant | compliant}",
      "afmTarget": "Protocol://{Host | Map}/Path"
      "afmAsyncDelay":" SecondsDelay",
      "afmDirLookupRefreshInterval": "SecondsInterval",
      "afmDirOpenRefreshInterval": "SecondsInterval",
      "afmEnableAutoEviction": "{yes | no}",
      "afmExpirationTimeout": "AFM expiration timeout",
      "afmFileLookupRefreshInterval": Seconds Interval,
      "afmMode": "{single-writer | sw | read-only | ro | local-updates | lu |
         independent-writer | Primary | Secondary}",
      "afmNumFlushThreads": NumThreads,
      "afmParallelReadChunkSize": NumBytes,
      "afmParallelReadThreshold": NumMegabytes,
      "afmParallelWriteChunkSize": NumBytes,
      "afmParallelWriteThreshold": NumMegabytes,
      "afmPrefetchThreshold": {1 | 1-99 | 100},
      "afmPrimaryID": "ID",
      "afmRPO": MinutesInterval,
      "afmShowHomeSnapshots": "{yes | no}",
}
```

Details of the parameters are given in the following list:

"filesetName": "*Fileset*",
The name of the fileset.

"path": "*Path*"

The path the fileset will be linked to. If not specified the fileset will be linked to the default path, for example, `/mnt/gpfs0/fset1`.

"owner": "*UserID*[:*GroupID*]"
The user ID of the owner of the new fileset, followed optionally by the group ID, as in "root:".

"permissions ": "*Integer*"
The permissions that are set using **chmod**. If not specified, a default ACL will be applied.

"inodeSpace": "*Inodes*"
The number of inodes that are allocated for use by the fileset.

"maxNumInodes": "*Inodes*"
The inode limit for the inode space that is owned by the specified fileset.

"permissionChangeMode": "*Mode*"
The permission change mode. Controls how `chmod` and ACL commands affect objects in the fileset.

chmodOnly
Only the `chmod` command can change access permissions.

setAclOnly
Only the ACL commands and API can change access permissions.

chmodAndSetAcl
Both the `chmod` command and ACL commands can change access permissions.

chmodAndUpdateAcl
Both the `chmod` command and ACL commands can change access permissions.

"comment": "*Comment*",
A comment that appears in the output of the `mmlsfileset` command.

"iamMode": "*Mode*"
The integrated archive manager (IAM) mode for the fileset.

advisory

noncompliant

compliant

"afmTarget": "*Home fileset*"
The home that is associated with the cache.

"afmAsyncDelay": ***Delay***
The time in seconds by which to delay write operations because of the lag in updating remote clusters.

"afmDirLookupRefreshInterval": ***Interval***
The interval in seconds between data revalidations of directories that are caused by lookup operations such as `ls` and `stat`.

"afmDirOpenRefreshInterval": ***Interval***
The interval in seconds between data revalidations of directories that are caused by lookup operations.

"afmEnableAutoEviction": "{yes | no}"
Enables or disables eviction on the fileset.

"afmExpirationTimeout": ***Interval***
The timeout in seconds after which cached data is considered expired. Used with `afmDisconnectTimeout`, which is set with `mmchconfig`.

"afmFileLookupRefreshInterval": ***Interval***
The interval in seconds between data revalidations of files that are caused by lookup operations such as `ls` and `stat`.

"afmMode": "{single-writer | sw | read-only | ro | local-updates | lu | independent-writer | Primary | Secondary}"
The mode in which the cache operates.

"afmNumFlushThreads": ***numThreads***
The number of threads that are used on each gateway to synchronize updates to the home cluster.

"afmParallelReadChunkSize": ***Size***
The minimum chunk size of read data, in bytes, that must be distributed among the gateway nodes during parallel reads.

"afmParallelReadThreshold": ***Threshold***
The minimum file size, in megabytes, at which to do parallel reads.

"afmParallelWriteChunkSize": ***Size***
The minimum chunk size of write data, in bytes, that must be distributed among the gateway nodes during parallel writes.

"afmParallelWriteThreshold": ***Threshold***
The minimum file size, in megabytes, at which to do parallel writes.

"afmPrefetchThreshold": {1 | 1-99 | 100}
The percentage of file size that must be cached before the entire file is prefetched.

0
Enables full file prefetching.

1-99
The percentage of file size that must be cached before the entire file is prefetched.

100
Disables full file prefetching.

"afmPrimaryID": "*ID***"**
The unique primary ID of the primary fileset for asynchronous data replication.

"afmRPO": ***Interval***
The recovery point objective (RPO) interval, in minutes, for a primary fileset.

"afmShowHomeSnapshots": "{yes | no}"
Shows or hides the home snapshot directory in cache. Specify `yes` to show, `no` to hide.

Response data

```
{
   "status": {
      "code":ReturnCode",
```

```
    "message":"ReturnMessage"
  },
  "jobs": [
     {
        "result":"",
          {
            "commands":"String",
            "progress":"String,
            "exitCode":"Exit code",
            "stderr":"Error",
            "stdout":"String",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted":"Time",
        "completed":"Time",
        "status":"Job status",
        }
  ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example creates and links a fileset `fset1` in file system `gpfs0`.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
-d '{
  "filesetName": "fset1",
  "path": "/mnt/gpfs0/fset1",
  "owner": "scalemgmt",
  "permissions": "700"
}' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets'
```

Response data:

```
{
  "jobs" : [ {
    "jobId" : 1000000000003,
    "status" : "COMPLETED",
    "submitted" : "2017-03-14 15:54:23,042",
    "completed" : "N/A",
    "request" : {
      "data" : {
        "filesetName" : "fset1",
        "owner" : "scalemgmt",
        "path" : "/mnt/gpfs0/fset1",
        "permissions" : 700
      },
      "type" : "POST",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/filesets"
    },
    "result" : { }
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing"
  }
}
```

"mmcrfileset command" on page 311
Creates a GPFS fileset.

"mmchfileset command" on page 224
Changes the attributes of a GPFS fileset.

"mmlsfileset command" on page 501
Displays attributes and status for GPFS filesets.

Filesystems/{filesystemName}/filesets/cos: POST

Creates a fileset that is related to a corresponding bucket.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `Filesystems/`*{filesystemName}*`/filesets/cos` request creates a fileset that is related to a corresponding bucket. For more information about the fields in the data structures that are returned, see “mmafmcosaccess command” on page 48.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/
{filesystemName}/filesets/cos
```

where

Filesystems/{filesystemName}/filesets/cos
Specifies the file system where the new fileset needs to be created.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 75. List of parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
  "filesetName": "COS fileset name",
  "endpoint": "URL of the object store",
  "useObjectFs": "true | false",
  "bucket": "Name of the bucket",
  "newBucket": "true | false",
  "dir": "Directory name",
  "policy": "Policy rules for the fileset",
  "tmpDir": "Directory pattern",
  "tmpFile": "File pattern",
  "quotaFiles": "Number",
  "quotaBlocks": "Number",
  "uid": "User ID",
```

```
   "gid": "Group ID",
   "permission": "Access permission",
   "mode": "Fileset access mode",
   "useUserKeys": "true | false",
   "chunkSize": "Chunk size",
   "readSize": "Read size",
   "useNoSubdir": "true | false",
   "useXattr": "true | false",
   "useSslCertVerify": "true | false",
   "useVhb": "true | false",
   "useGcs": "true | false",
   "accessKey": "Access key",
   "secretKey": "Secret key"
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"filesetName": "*COS fileset name*"
Name of the Cloud Object Storage fileset.

"endpoint": "*URL of the object store*"
URL to your object store. Use server's hostname or IP or name of the map.

"useObjectFs": "*true | false*"
Whether to handle Cloud Object Storage more like a file system.

"bucket": "*Name of the bucket*"
Name of the bucket that is related to your fileset. You can skip this parameter if your fileset has the same name as your bucket.

"newBucket": "*true | false*"
Whether to create a Cloud Object Storage bucket.

"dir": "*Directory name*"
Name of the directory name or full path inside a file system, where you need to link the fileset. If you skip this parameter, the fileset name will be used as the directory name.

"policy": "*Policy rules for the fileset*"
Policy rules for the fileset.

"tmpDir": "*Directory pattern*"
Directory pattern to keep local, if no policy is specified.

"tmpFile": "*File pattern*"
File pattern to keep local, if no policy is specified.

"quotaFiles": "*Number*"
Quota for the number of files. Enables eviction when 80% of the files are used.

"quotaBlocks": "*Number*"
Quota limit for the blocks. Enables eviction when 80% of the quotas are used.

"uid": "*User ID*"
User ID of the fileset owner.

"gid": "*Group ID*"
Group ID of the fileset users.

"permission": "*Access permission*"
Access permissions in octal format.

"mode": "*Fileset access mode*"
Fileset access mode. Possible values are: `iw` (independent-writer), `sw` (single-writer), `ro` (read-only), or `lu` (local-updates)

"useUserKeys": "*true | false*"
Whether to use user keys on requests from object store.

"chunkSize": "*Chunk size*",
Chunk size to control number of upload multiple parts.

"readSize": "*Read size*"
Download size [in numeric], default is zero to get the full object.

"useNoSubdir": "*true | false*"
Whether to create a subdirectory if '/' in object name.

"useXattr": "*true | false*"
Whether to get or set user extended attributes.

"useSslCertVerify": "*true | false*"
Whether to enable SSL certificate verification, valid only with HTTPS.

"useVhb": "*true | false*"
Whether to use Virtual Hosted Bucket, for Alibaba OSS.

"useGcs": "*true | false*"
Whether to use Google Cloud Services, for Google GCS.

"accessKey": "*Access key*"
Access key for your bucket. Use together with Secret key.

"secretKey": "*Secret key*"
Secret key for your bucket. Use together with Access key.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success. A nonzero value denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example creates a new fileset that is related to a corresponding bucket.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json' -d '{ \
   "filesetName": "mycosfset", \
   "endpoint": "http://s3store.com:8080", \
   "useObjectFs": true, \
   "bucket": "mybucket", \
   "newBucket": "false", \
   "dir": "mycosfset", \
   "policy": "[ RULE1, RULE2 ]", \
   "tmpDir": "dirpattern", \
   "tmpFile": "%.log", \
   "quotaFiles": 1000, \
   "quotaBlocks": 20, \
   "uid": "994", \
   "gid": "992", \
   "permission": "0770", \
   "mode": "sw", \
   "useUserKeys": false, \
   "chunkSize": 5, \
   "readSize": 3, \
   "useNoSubdir": false, \
   "useXattr": false, \
   "useSslCertVerify": false, \
   "useVhb": false, \
   "useGcs": false, \
   "accessKey": "ACCESS_KEY", \
   "secretKey": "SECRET_KEY" \
 }' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/cos'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.


```
{
  "jobs" : [ {
    "jobId" : 1000000000012,
    "status" : "FAILED",
    "submitted" : "2020-09-29 15:57:42,926",
    "completed" : "2020-09-29 15:57:50,251",
    "runtime" : 7325,
    "request" : {
      "data" : {
        "accessKey" : "****",
        "bucket" : "mybucket",
        "chunkSize" : 5,
        "dir" : "mycosfset",
        "endpoint" : "http://s3store.com:8080",
        "filesetName" : "mycosfset",
        "gid" : "992",
        "mode" : "sw",
        "permission" : "0770",
        "quotaBlocks" : 20,
        "quotaFiles" : 1000,
        "readSize" : 3,
        "secretKey" : "****",
        "tmpDir" : "dirpattern",
        "tmpFile" : "%.log",
        "uid" : "994",
        "useGcs" : false,
        "useNoSubdir" : false,
        "useObjectFs" : true,
        "useSslCertVerify" : false,
        "useUserKeys" : false,
        "useVhb" : false,
        "useXattr" : false
      },
      "type" : "POST",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/filesets/cos"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmafmcoskeys 'mybucket' set  ****  **** ", "mmafmcosconfig 'gpfs0'
'mycosfset' --endpoint 'http://s3store.com:8080' --object-fs --bucket 'mybucket' --dir
'mycosfset' --tmpdir 'dirpattern' --tmpFile '%.log' --quota-files 1000 --quota-blocks 20 --uid
'994' --gid '992' --perm '0770' --mode 'sw'   --chunk-size 5 --read-size 3           " ],
      "stdout" : [ ],
      "stderr" : [ "EFSSG0053C Failed to execute command:\nmmafmcosconfig: Creating fileset
mycosfset failed. Check logs for details.\nmmafmcosconfig: Command failed. Examine previous
error messages to determine cause.\n." ],
      "exitCode" : 8
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```


Related reference

“mmafmcosaccess command” on page 48

Maps a directory in an AFM to cloud object storage fileset to the bucket on a cloud object storage.

Filesystems/{filesystemName}/filesets/{filesetName}: DELETE

Deletes a specific fileset.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `DELETE filesystems/{filesystemName}/filesets/{filesetName}` command deletes the specified fileset. For more information on deleting snapshots, see “mmdelfileset command” on page 369.

Request URL

Use this URL to delete a global snapshot:

```
https://management API host:port/scalemgmt/v2/filesystems/filesystemName/filesets/filesetName
```

where:

filesystems/filesystemName
Specifies the file system to which the fileset belongs. Required.

filesets/filesetName
Specifies the fileset to be deleted. Required.

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 76. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
filesetName	The fileset name. This is the path of the fileset.	Required.
qosClass	The QoS class in which the operation is performed.	Optional.

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
```

```
            "stderr":"Error",
            "stdout":"String",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted":"Time",
        "completed":Time",
        "status":"Job status",
        }
      ],
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example deletes the fileset *myFset1* from the file system *gpfs0*.

Request data:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": ["'mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": ["''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": ["''EFSSG0740C There are not enough resources available to create a
                  new independent file set.'',  ...]",
        "stdout": ["''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "DELETE",
        "url": "/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1",
        "data": "",
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

See also

- “mmdelfileset command” on page 369

Filesystems/{filesystemName}/filesets/{filesetName}: GET

Gets information about a fileset.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET filesystems/{filesystemName}/filesets/{filesetName}` request gets information about the specified fileset in a file system. For more information about the fields in the data structures that are returned, see the topics "mmcrfileset command" on page 311, "mmchfileset command" on page 224, and "mmlsfileset command" on page 501.

Request URL

```
https://<IP or host name of API server>:<port>/scalemgmt/v2/filsystems/filesystemName/filesets/
filesetName
```

where:

filesystem/filesystemName
Specifies the name of the file system to which the filesets belong. Required.

filesets/filesetName
Specifies the fileset about which you need information. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 77. List of parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
filesetName	The fileset name. This is the path of the fileset.	Required.
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.

Request data

No request data.

Response data

```
{
  "status": {
```

```
      "code":ReturnCode,
      "message":"ReturnMessage"
   },
"paging":
 {
   "next": "URL",
   "fields": "Fields",
   "filter": "Filters",
   "baseUrl": "Base URL",
   "lastId": "ID of the last element",
 },
  "filesets": [
     {
           "filesetName":"Fileset",
           "filesystemName":"Device",
           "config":
            {
              "path": "Path",
              "inodeSpace":"Inodes",
              "maxNumInodes":"Inodes",
              "permissionChangeMode":"Mode",
              "comment":"Comment",
              "iamMode":"Mode",
              "oid": "ID",
              "id":"ID",
              "status":"Status"
              "parentId":"ID",
              "created":"DateTime",
              "isInodeSpaceOwner":"isOwner",
              "inodeSpaceMask":"inodeSpace",
              "snapID":"ID",
              "rootInode":"Inode",
         }
       "afm":
         {
           "afmTarget":"Protocol://{Host | Map}/Path",
           "afmState":"enabled | disabled",
           "afmMode": "{single-writer | sw | read-only | ro | local-updates | lu |
              independent-writer | Primary | Secondary}",
           "afmFileLookupRefreshInterval":"Interval",
           "afmDirLookupRefreshInterval":"Interval",
           "afmDirOpenRefreshInterval":"Interval",
           "afmAsyncDelay":"Delay",
           "afmNeedsRecovery":"true | false"
           "afmExpirationTimeout":"Interval",
           "afmRPO":"Interval",
           "afmLastPSnapId":"Psnap ID"
           "afmShowHomeSnapshots":"{yes | no}",
           "afmNumReadThreads":"Number of read threads",
           "afmReadBufferSize":"Read buffer size",
           "afmWriteBufferSize":"Write buffer size",
           "afmReadSparseThreshold":"Read sparse thread",
           "afmParallelReadChunkSize":"Size",
           "afmParallelReadThreshold":"Threshold",
           "afmNumFlushThreads":"Threads",
           "afmPrefetchThreshold":"{1 | 1-99 | 100}",
           "afmEnableAutoEviction":"{yes | no}",
           "afmParallelWriteThreshold":"Threshold",
           "afmNeedsResync":"true | false",
           "afmParallelWriteChunkSize":"Size",
           "afmNumWriteThreads":"Number of write threads",
           "afmPrimaryID":"ID",
           "afmDRState":"DR state"
           "afmAssociatedPrimaryId":"ID"
           "afmDIO":"",
           "afmHostMapName":"Name",
           "afmHostMapping":"Mapping details"[
            {
             "home":"IP or host name of home",
             "cache":"IP or hostname of cache",
            }
           ]
          "afmGatewayNode":"IP or host name of home",
          "afmIOFlags":"I/O flag",
          "afmVerifyDmapi":"true | false",
          "afmSkipHomeACL":"true | false",
          "afmSkipHomeMtimeNsec":"true | false",
          "afmForceCtimeChange":"true | false",
          "afmSkipResyncRecovery":"true | false",
          "afmSkipConflictQDrop":"true | false",
          "afmRefreshAsync":"true | false",
          "afmParallelMounts":"true | false",
```

```
            "afmRefreshOnce":"true | false",
            "afmSkipHomeCtimeNsec":"true | false",
            "afmReaddirOnce":"true | false",
            "afmResyncVer2":"true | false",
            "afmSnapUncachedRead":"true | false",
            "afmFastCreate":"true | false",
            "afmObjectXattr":"true | false",
            "afmObjectVHB":"true | false",
            "afmObjectBlkIO":"true | false",
            "afmSkipHomeRefresh":"true | false",
            "afmObjectGCS":"true | false",
            "afmObjectUserKeys":"true | false",
            "afmWriteOnClose":"true | false",
            "afmObjectSubdir":"true | false",
            "afmObjectSSL":"true | false",
            "cacheState":"State of the cache",
            "queueLength":"Queue length",
            "queueNumExec":"Queue number",
            "afmHostMaps":"Host map details",[
              {
              "home":"IP address or hostname of the home",
              "cache":"IP address or hostname of the cache",
              }
              ]
        },
    "usage":
      {
       "usedInodes":"Inodes used",
       "allocatedInodes":"Inodes allocated",
       "usedBytes":"Used capacity",
       "inodeSpaceUsedInodes":"Inode space used",
       "inodeSpaceFreeInodes":"Inode space available",
      }
}
}
```

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code":"*ReturnCode*"
The return code.

"paging":
Paging details.

"next": "*URL*",
The URL to retrieve the next page. Paging is enabled when more than 1000 objects would be returned by the query.

"fields": "*Fields*",
The fields used in the original request.

"filter": "*Filters*",
The filter used in the original request.

"baseUrl": "*Base URL*",
The URL of the request without any parameters.

"lastId": "*ID of the last element*",
The ID of the last element that can be used to retrieve the next elements.

"*filesets*":
An array of information about the filesets in the specified file system. Each array element describes one fileset and can contain the data structures `afm`, `config`, `links`, and `state`. For more information about the fields in these data structures, see the links at the end of this topic:

"filesets":
Information about the fileset configuration.

"filesetName": "*Fileset*",
The name of the fileset.

"filesystemName": "*Device*"
Required. The file system in which the fileset is located.

"config":

"path": "*Path*"
The absolute path of the fileset.

"inodeSpace": "*Inodes*"
The number of inodes that are allocated for use by the fileset.

"maxNumInodes": "*Inodes*"
The inode limit for the inode space that is owned by the specified fileset.

"permissionChangeMode": "*Mode*"
The permission change mode. Controls how chmod and ACL commands affect objects in the fileset.

chmodOnly
Only the chmod command can change access permissions.

setAclOnly
Only the ACL commands and API can change access permissions.

chmodAndSetAcl
Both the chmod command and ACL commands can change access permissions.

chmodAndUpdateAcl
Both the chmod command and ACL commands can change access permissions.

"comment": "*Comment*",
A comment that appears in the output of the `mmlsfileset` command.

"iamMode": "*Mode*"
The integrated archive manager (IAM) mode for the fileset.

ad | advisory

nc | noncompliant

co | compliant

"oid": "*ID*"
Internal identifier of the fileset that is used for paging.

"id": *ID*
The fileset identifier.

"status": "*Status*"
Specifies whether the file is linked or unlinked.

"parentId": *ParentID*
The parent identifier of the fileset.

"created": "*DateTime*",
The date and time when the fileset was created.

"isInodeSpaceOwner": "*isOwner*",
Indicates whether the fileset has its own inode space.

"isInodeSpaceMask": "*isMask*",
The inode space mask.

"snapID": *ID*
The snapshot ID.

"afm":
Information about Active File Management (AFM).

"afmTarget": "*Interval*"*Protocol*://{Host | Map}/Path"
The home that is associated with the cache.

"afmState":"*enabled | disabled*"

"afmMode": "{single-writer | sw | read-only | ro | local-updates | lu | independent-writer | Primary | Secondary}"
The mode in which the cache operates.

"afmFileLookupRefreshInterval": *Interval*
The interval in seconds between data revalidations of files that are caused by lookup operations such as `ls` and `stat`.

"afmDirLookupRefreshInterval":" *Interval*"
The interval in seconds between data revalidations of directories that are caused by lookup operations such as `ls` and `stat`.

"afmDirOpenRefreshInterval": *Interval*
The interval in seconds between data revalidations of directories that are caused by lookup operations.

"afmAsyncDelay": "*Delay*"
The time in seconds by which to delay write operations because of the lag in updating remote clusters.

"afmNeedsRecovery":"*true | false*"

"afmExpirationTimeout": *Interval*
The timeout in seconds after which cached data is considered expired. Used with `afmDisconnectTimeout`, which is set with `mmchconfig`.

"afmRPO": *Interval*
The recovery point objective (RPO) interval, in minutes, for a primary fileset.

"afmLastPSnapId":"*Psnap ID*"

"afmShowHomeSnapshots": "{yes | no}"
Shows or hides the home snapshot directory in cache. Specify `yes` to show, `no` to hide.

"afmNumReadThreads":"*Number of read threads*"
Number of AFM read threads

"afmReadBufferSize":"*Read buffer size*"
AFM read buffer size.

"afmWriteBufferSize":"*Write buffer size*"
AFM write buffer size.

"afmReadSparseThreshold":"*Read sparse thread*"
Read sparse threshold.

"afmParallelReadChunkSize": *Size*
The minimum chunk size of read data, in bytes, that must be distributed among the gateway nodes during parallel reads.

"afmParallelReadThreshold": *Threshold*
The minimum file size, in megabytes, at which to do parallel reads.

"afmNumFlushThreads": *numThreads*
The number of threads that are used on each gateway to synchronize updates to the home cluster.

"afmPrefetchThreshold": {1 | 1-99 | 100}
The percentage of file size that must be cached before the entire file is prefetched.

0
Enables full file prefetching.

1-99
The percentage of file size that must be cached before the entire file is prefetched.

100
Disables full file prefetching.

"afmEnableAutoEviction": "{yes | no}"
Enables or disables eviction on the fileset.

"afmParallelWriteThreshold": ***Threshold***
The minimum file size, in megabytes, at which to do parallel writes.

"afmNeedsResync":"*true | false*"
Whether the AFM needs resynchronization.

"afmParallelWriteChunkSize": ***Size***
The minimum chunk size of write data, in bytes, that must be distributed among the gateway nodes during parallel writes.

"afmNumWriteThreads":"*Number of write threads*"

"afmPrimaryID": "*ID*"
The unique primary ID of the primary fileset for asynchronous data replication.

afmDRState":"*DR state*"
Status of the AFM DR.

"afmAssociatedPrimaryId":"*ID*"
Associated primary ID.

"afmDIO":"*Number*"
IO details.

"afmHostMapName":"*Name*"
The name of the AFM host map this fileset belongs to

"afmHostMapping":

"home":"*Home address*"
IP or hostname of home.

"cache":"*Cache address*"
IP or hostname of cache

"afmGatewayNode":"*Gateway node*",
Name of the gateway node.

"afmIOFlags":"*I/O flag*"
AFM I/O flag.

"afmVerifyDmapi":"*true | false*"
Whether to verify AFM DMAPI.

"afmSkipHomeACL":"*true | false*"
Whether to ignore home ACL.

"afmSkipHomeMtimeNsec":"*true | false*"
Whether to skip home *mtime* and *nsec*.

"afmForceCtimeChange":"*true | false*"
Whether to force change *ctime*.

"afmSkipResyncRecovery":"*true | false*"

"afmSkipConflictQDrop":"*true | false*"

"afmRefreshAsync":"*true | false*"

"afmParallelMounts":"*true | false*"

"afmRefreshOnce":"*true | false*"

"afmSkipHomeCtimeNsec":"*true | false*"

"afmReaddirOnce":"*true | false*"

"afmResyncVer2":"*true | false*"

"afmSnapUncachedRead":"*true | false*"

"afmFastCreate":"*true | false*"

"afmObjectXattr":"*true | false*"
Whether user extended attributes are available in object store.

"afmObjectVHB":"*true | false*"
Whether to use Virtual Hosted Bucket, for Alibaba Clod Object Storage Service (OSS).

"afmObjectBlkIO":"*true | false*"

"afmSkipHomeRefresh":"*true | false*"

"afmObjectGCS":"*true | false*"
Whether to use Google Cloud Services.

"afmObjectUserKeys":"*true | false*"
Whether to use user keys on requests from the object store.

"afmWriteOnClose":"*true | false*"

"afmObjectSubdir":"*true | false*"
Whether to create sub-directory if '/' is present in the object name.

"afmObjectSSL":"*true | false*"
Whether to enable SSL certificate verification, which is valid only with HTTPS from the object store.

"cacheState":"*State of the cache*"
Current cache state.

"queueLength":""
Length of the queue on the primary gateway.

"queueNumExec":""
Number of operations played at home since the fileset is last active.

"afmHostMaps":""

"home":""
IP address or hostname of the home site.

"cache":""
IP address or hostname of the cache site.

"usage":""

"usedInodes":"*Used inodes*"
Specifies the used inodes for the fileset.

"allocatedInodes":"*allocated inodes*"
Specifies the allocated inodes for the fileset.

"usedBytes":"*Used bytes*"
Specifies the used bytes for the fileset.

"inodeSpaceUsedInodes":"*Used inode space*"
Specifies the used inode of the fileset.

"inodeSpaceFreeInodes":"*Free inodes*"
Specifies the number of inode that are available for the fileset.

Examples

The following example gets information about the fileset `myFset1` in file system `gpfs0`.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/myfset1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
```

```
  "code": 200,
  "message": "..."
},
"paging": {
  "next": "https://localhost:443/scalemgmt/v2/filesystems/gpfs0/filesets?lastId=10001",
  "fields": "period,restrict,sensorName",
  "filter": "usedInodes>100,maxInodes>1024",
  "baseUrl": "/scalemgmt/v2/perfmon/sensor/config",
  "lastId": 10001
},
"filesets": [
  {
    "filesetName": "myFset1",
    "filesystemName": "gpfs0",
    "config": {
      "path": "/mnt/gpfs0/myFset1",
      "inodeSpace": 0,
      "maxNumInodes": 4096,
      "permissionChangeMode": "chmodAndSetAcl",
      "comment": "Comment1",
      "iamMode": "off",
      "oid": 158,
      "id": 5,
      "status": "Linked",
      "parentId": 1,
      "created": "2016-12-13 13.59.15",
      "isInodeSpaceOwner": false,
      "inodeSpaceMask": 0,
      "snapId": 0,
      "rootInode": 131075
    },
    "afm": {
      "afmTarget": "nfs://198.51.100.10/gpfs/afmHomeFs/afmHomeFileset2",
      "afmState": "enabled",
      "afmMode": "read-only",
      "afmFileLookupRefreshInterval": 30,
      "afmFileOpenRefreshInterval": 0,
      "afmDirLookupRefreshInterval": 60,
      "afmDirOpenRefreshInterval": 60,
      "afmAsyncDelay": 0,
      "afmNeedsRecovery": true,
      "afmExpirationTimeout": 100,
      "afmRPO": 0,
      "afmLastPSnapId": 0,
      "afmShowHomeSnapshots": false,
      "afmNumReadThreads": 0,
      "afmReadBufferSize": 0,
      "afmWriteBufferSize": 0,
      "afmReadSparseThreshold": 0,
      "afmParallelReadChunkSize": 134217728,
      "afmParallelReadThreshold": 1024,
      "afmNumFlushThreads": 4,
      "afmPrefetchThreshold": 0,
      "afmEnableAutoEviction": false,
      "afmParallelWriteThreshold": 0,
      "afmNeedsResync": true,
      "afmParallelWriteChunkSize": 0,
      "afmNumWriteThreads": 4,
      "afmPrimaryID": "string",
      "afmDRState": "string",
      "afmAssociatedPrimaryId": "string",
      "afmDIO": 0,
      "afmHostMapName": "map1",
      "afmHostMapping": [
        {
          "home": "10.0.100.13",
          "cache": "cache-41"
        }
      ],
      "afmGatewayNode": "testnode-11",
      "afmIOFlags": "0x4000000",
      "afmVerifyDmapi": true,
      "afmSkipHomeACL": true,
      "afmSkipHomeMtimeNsec": true,
      "afmForceCtimeChange": true,
      "afmSkipResyncRecovery": true,
      "afmSkipConflictQDrop": true,
      "afmRefreshAsync": true,
      "afmParallelMounts": true,
      "afmRefreshOnce": true,
      "afmSkipHomeCtimeNsec": true,
      "afmReaddirOnce": true,
```

```
        "afmResyncVer2": true,
        "afmSnapUncachedRead": true,
        "afmFastCreate": true,
        "afmObjectXattr": false,
        "afmObjectVHB": false,
        "afmObjectBlkIO": true,
        "afmSkipHomeRefresh": true,
        "afmObjectGCS": false,
        "afmObjectUserKeys": false,
        "afmWriteOnClose": true,
        "afmObjectSubdir": true,
        "afmObjectSSL": false,
        "cacheState": "Active",
        "queueLength": 5,
        "queueNumExec": 10,
        "afmHostMaps": [
          {
            "home": "10.0.100.13",
            "cache": "cache-41"
          }
        ]
      },
      "usage": {
        "usedInodes": 1000,
        "allocatedInodes": 65856,
        "usedBytes": 134217728,
        "inodeSpaceUsedInodes": 1100,
        "inodeSpaceFreeInodes": 55856
      }
    }
  ]
}
```


“mmcrfileset command” on page 311
Creates a GPFS fileset.

“mmchfileset command” on page 224
Changes the attributes of a GPFS fileset.

“mmlsfileset command” on page 501
Displays attributes and status for GPFS filesets.

Filesystems/{filesystemName}/filesets/{filesetName}: PUT

Modifies a fileset.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `filesystems/{filesystemName}/filesets/{filesetName}` request modifies a specific fileset, which is part of the specified file system. For more information about the fields in the data structures that are returned, see the topics "mmcrfileset command" on page 311, "mmchfileset command" on page 224, and "mmlsfileset command" on page 501.

Request URL

```
https://<IP or host name of API server>:port/scalemgmt/v2/filesystems/filesystemName/filesets/
filesetName
```

where:

filesystems/filesystemName
Specifies the name of the file system to which the fileset belongs. Required.

filesets/filesetName
Specifies fileset to be modified. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 78. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
filesetName	The fileset name. This is the path of the fileset.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
      "newFilesetName": "Fileset",
      "maxNumInodes": "Inodes",
      "allocInodes": "Inodes",
```

```
      "permissionChangeMode": "chmodOnly | setAclOnly | chmodAndSetAcl",
      "comment": "Comment",
      "iamMode": "{advisory | noncompliant | compliant}",
      "afmTarget": "Protocol://{Host | Map}/Path"
      "afmAsyncDelay": SecondsDelay,
      "afmDirLookupRefreshInterval": SecondsInterval,
      "afmDirOpenRefreshInterval": SecondsInterval,
      "afmEnableAutoEviction": "{yes | no}",
      "afmExpirationTimeout": SecondsInterval,
      "afmFileLookupRefreshInterval": SecondsInterval,
      "afmMode": "{single-writer | sw | read-only | ro | local-updates | lu |
         independent-writer | Primary | Secondary}",
      "afmNumFlushThreads": NumThreads,
      "afmParallelReadChunkSize": NumBytes,
      "afmParallelReadThreshold": NumMegabytes,
      "afmParallelWriteChunkSize": NumBytes,
      "afmParallelWriteThreshold": NumMegabytes,
      "afmPrefetchThreshold": {1 | 1-99 | 100},
      "afmPrimaryID": "ID",
      "afmRPO": MinutesInterval,
      "afmShowHomeSnapshots": "{yes | no}",
      "afmObjectSubdir":"true | false",
            "afmObjectUserKeys":"true | false",
            "afmObjectXattr":"true | false",
            "afmObjectSSL":"true | false",
            "afmObjectVHB":"true | false",
            "afmObjectGCS":"true | false",

 }
```

```
}
```

"newFilesetName": "*Fileset*",
The name of the fileset.

"maxNumInodes": "*Inodes*"
The inode limit for the inode space that is owned by the specified fileset.

"permissionChangeMode": "*Mode*"
The permission change mode. Controls how chmod and ACL commands affect objects in the fileset.

chmodOnly
Only the chmod command can change access permissions.

setAclOnly
Only the ACL commands and API can change access permissions.

chmodAndSetAcl
Both the chmod command and ACL commands can change access permissions.

chmodAndUpdateAcl
Both the chmod command and ACL commands can change access permissions.

"comment": "*Comment*",
A comment that appears in the output of the `mmlsfileset` command.

"iamMode": "*Mode*"
The integrated archive manager (IAM) mode for the fileset.

advisory

noncompliant

compliant

"afmTarget": *Interval*"*Protocol*://{Host | Map}/Path"
The home that is associated with the cache.

"afmAsyncDelay": *Delay*
The time in seconds by which to delay write operations because of the lag in updating remote clusters.

"afmDirLookupRefreshInterval": ***Interval***
The interval in seconds between data revalidations of directories that are caused by lookup operations such as `ls` and `stat`.

"afmDirOpenRefreshInterval": ***Interval***
The interval in seconds between data revalidations of directories that are caused by lookup operations.

"afmEnableAutoEviction": "{yes | no}"
Enables or disables eviction on the fileset.

"afmExpirationTimeout": ***Interval***
The timeout in seconds after which cached data is considered expired. Used with `afmDisconnectTimeout`, which is set with `mmchconfig`.

"afmFileLookupRefreshInterval": ***Interval***
The interval in seconds between data revalidations of files that are caused by lookup operations such as `ls` and `stat`.

"afmMode": "{single-writer | sw | read-only | ro | local-updates | lu | independent-writer | Primary | Secondary}"
The mode in which the cache operates.

"afmNumFlushThreads": ***numThreads***
The number of threads that are used on each gateway to synchronize updates to the home cluster.

"afmParallelReadChunkSize": ***Size***
The minimum chunk size of read data, in bytes, that must be distributed among the gateway nodes during parallel reads.

"afmParallelReadThreshold": ***Threshold***
The minimum file size, in megabytes, at which to do parallel reads.

"afmParallelWriteChunkSize": ***Size***
The minimum chunk size of write data, in bytes, that must be distributed among the gateway nodes during parallel writes.

"afmParallelWriteThreshold": ***Threshold***
The minimum file size, in megabytes, at which to do parallel writes.

"afmPrefetchThreshold": {1 | 1-99 | 100}
The percentage of file size that must be cached before the entire file is prefetched.

0
Enables full file prefetching.

1-99
The percentage of file size that must be cached before the entire file is prefetched.

100
Disables full file prefetching.

"afmPrimaryID": "*ID*"
The unique primary ID of the primary fileset for asynchronous data replication.

"afmRPO": ***Interval***
The recovery point objective (RPO) interval, in minutes, for a primary fileset.

"afmShowHomeSnapshots": "{yes | no}"
Shows or hides the home snapshot directory in cache. Specify `yes` to show, `no` to hide.

"afmObjectSubdir":"*true | false*"
Whether to create sub-directory if '/' is present in the object name.

"afmObjectUserKeys":"*true | false*"
Whether to use user keys on requests from the object store.

"afmObjectXattr":"*true | false*"
Whether user extended attributes are available in object store.

"afmObjectSSL":"*true | false*"
Whether to enable SSL certificate verification, which is valid only with HTTPS from the object store.

"afmObjectVHB":"*true | false*"
Whether to use Virtual Hosted Bucket, for Alibaba Clod Object Storage Service (OSS).

"afmObjectGCS":"*true | false*"
Whether to use Google Cloud Services.

Response data

```
{
     "status": {
       "code":ReturnCode",
       "message":"ReturnMessage"
    },
    jobs: [
       {
          "result":"",
            {
              "commands":"String",
              "progress":"String,
              "exitCode":"Exit code",
              "stderr":"Error",
              "stdout":"String",
            },
          "request":" ",
            {
              "type":"{GET | POST | PUT | DELETE}",
              "url":"URL",
              "data":"",
            }
          "jobId":"ID",
          "submitted":"Time",
          "completed":Time",
          "status":"Job status",
          }
        ],
    }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example gets information about the fileset `myFset1` in file system `gpfs0`.

Request data:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
-d '{
  "newFilesetName": "newFset1",
  "maxNumInodes": "100M",
  "allocInodes": "5M",
  "permissionChangeMode": "chmodAndSetAcl",
  "comment": "New comment",
  "iamMode": "compliant",
  "afmTarget": "nfs://198.51.100.10/gpfs/afmHomeFs/afmHomeFileset2",
  "afmAsyncDelay": 0,
  "afmDirLookupRefreshInterval": 60,
  "afmDirOpenRefreshInterval": 60,
  "afmEnableAutoEviction": false,
  "afmExpirationTimeout": 100,
  "afmFileLookupRefreshInterval": 30,
  "afmMode": "read-only",
  "afmNumFlushThreads": 4,
  "afmParallelReadChunkSize": 134217728,
  "afmParallelReadThreshold": 1024,
  "afmParallelWriteChunkSize": 0,
  "afmParallelWriteThreshold": 0,
  "afmPrefetchThreshold": 0,
  "afmPrimaryID": "string",
  "afmRPO": 0,
  "afmShowHomeSnapshots": false,
  "afmObjectSubdir": true,
  "afmObjectUserKeys": false,
  "afmObjectXattr": false,
  "afmObjectSSL": false,
  "afmObjectVHB": false,
  "afmObjectGCS": false
}' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/fileset1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000007,
    "status" : "FAILED",
    "submitted" : "2020-09-29 15:17:52,389",
    "completed" : "2020-09-29 15:17:53,388",
```

```
    "runtime" : 999,
    "request" : {
      "data" : {
        "afmObjectGCS" : false,
        "afmObjectSSL" : false,
        "afmObjectSubdir" : true,
        "afmObjectUserKeys" : false,
        "afmObjectVHB" : false,
        "afmObjectXattr" : false
      },
      "type" : "PUT",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/filesets/fileset1"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmchfileset 'gpfs0' 'fileset1' -p 'afmObjectSubdir=yes' -p
'afmObjectUserKeys=no' -p 'afmObjectXattr=no' -p 'afmObjectSSL=no' -p 'afmObjectVHB=no' -p
'afmObjectGCS=no' " ],
      "stdout" : [ ],
      "stderr" : [ "EFSSG0053C Failed to execute command:Incorrect fileset name
fileset1.\nmmchfileset: Command failed. Examine previous error messages to determine
cause.\n." ],
      "exitCode" : 8
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

See also

- “mmchfileset command” on page 224
- “mmcrfileset command” on page 311
- “mmlsfileset command” on page 501

Filesystems/{filesystemName}/filesets/{filesetName}/afmctl: POST

Sets the control values for Active File Management (AFM).

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `filesystems/{filesystemName}/filesets/{filesetName}/afmctl` command defines the attributes that control the AFM function of a specific fileset. For more information about the fields in the request data structures, see the topics "mmafmctl command" on page 61, "mmafmconfig command" on page 45, and "mmafmlocal command" on page 78.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/{filesystemName}
```

```
/filesets/{filesetName}/afmctl
```

where

filesystem/filessystemName
Specifies the file system in which the fileset is present. Required.

filesets/filesetName/afmctl
Specifies the fileset for which the AFM controls are to be defined. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 79. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
filesetName	The fileset name. This is the path of the fileset.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
 {
          "failback":
          {
          "action": "Start | Stop",
          "failoverTime": "Failover time",
          },
        "failoverToSecondary": .,
          {
          "restore": "Restore type",
          },
          "failover": {
    "newTarget": "Target URL ",
    "targetOnly": true | false
          },
        "convertToPrimary":
          {
          "afmtarget": "Target fileset",
          "inband": "Inband",
          "secondarySnapName": "Secondary snap name",
          "noCheckMetadata": "Yes | No",
          "rpo": "RPO interval",
          },
"convertToSecondary":
          {
          "primaryid": "ID",
          "force": "Yes | No",
          },
"changeSecondary":
          {
          "newTarget": "New target fileset",
          "inband": "Inband",
          "targetOnly": "Target-only",
          },
"replacePrimary": "Replace primary fileset",
"failbackToPrimary":
          {
          "action": "Start | Stop",
          "force": "Yes | No",
          },
 "applyUpdates": "Apply updates",
 "prefetch": {
  "policy": True | False,
  "gatewayNode":"Name of the Node",
  "getnThreads":Integer value,
  "force ":True |False,
  "enableFailedFileList":True | False,
  "metaDataOnly":True |False,
  "listFile": "File path location",
  "dirListFile": "File path location",
  "homeFileSystemPath": "File path location
  },
  "evict": {
    "safeLimit": "The safe limit",
    "fileName": "The name of the file",
    "logFile ": " Eviction Log file name,
    "listfile": "File name",
    "filePath": "location of file to be evicted"
 },
 "calledSetterCount": "Called setter count",
 }
```

Details of the parameters are given in the following list:

"failback":

"action": "*Start | Stop*"
The failback action to be performed.

"failoverTime": "*Failover time*"
Specifies the failover time.

"failoverToSecondary":

"restore": "*Restore type*"
Specifies the restore type.

"failover"

newTarget : "*Target URL*"
Specifies the name and path of the new home server. This path replaces the server details that were originally defined in the **afmtarget** parameter of the **mmcrfileset** command.

"targetOnly" : *true | false*
Specifies whether the mount path or IP address of the target path is changed. Do not use this option to change the target location or the protocol. The new NFS server must be in the same home cluster and must display the same architecture as the existing NFS server in the target path.

"convertToPrimary":

"afmtarget": "*Target fileset*"
Target fileset for the AFM operation.

"inband": "*Inband*"
Used for inband trucking. Inband trucking is the process of copying the data while the primary-secondary relationship from GPFS fileset is set up, where the primary site has contents and the secondary site is empty.

"secondarySnapName": "*Secondary snap name*"
Used while a new primary is established for an existing secondary or acting as the primary during failback.

"noCheckMetadata": "*Yes | No*"
Used if one needs to proceed with conversion without checking for append-only or immutable files.

"rpo": "*RPO interval*"
Specifies the RPO interval in minutes for the primary fileset.

"convertToSecondary":

"primaryid": "*ID*",
Specifies the ID of the primary with which the secondary is associated.

"force": "*Yes | No*",
If **convertToSecondary** failed or got interrupted, it does not create **afmctl** file at the secondary. In such scenarios, rerun the command with the **--force** option.

"changeSecondary":

"newTarget": "*New target fileset*"
Specifies a new home server and path, replacing the home server and path that is originally set.

"inband": "*Inband*"
Used for inband trucking. Inband trucking is the process of copying the data while a primary-secondary relationship from GPFS fileset is defined, where the primary site has contents and the secondary site is empty.

"targetOnly": "*Target-only*"
Used when you want to change the IP address or NFS server name for the same target path. The new NFS server must be in the same home cluster and must be of the same architecture (power or x86) as the existing NFS server in the target path. This option can be used to move from NFS to a mapping target.

"replacePrimary": "*Replace primary fileset*"
Replace the primary fileset in the AFM relationship.

"failbackToPrimary":

"action": "*Start | Stop*",
The failback action to be carried out.

"force": "*Yes | No*",
Used if stop or start does not complete successfully due to any errors, and if it does not allow *failbackToPrimary* action to stop or start again.

"applyUpdates": "*Apply updates*",
Whether to apply updates.

"prefetch"
Prefetch the list of files from the home cluster.

"policy":*True | False*
Specifies whether a GPFS policy is used to generate the list file. The policy helps sequences like '\' or '\n' to be escaped as '\' and '\n' .

"gatewayNode":*"Node name"*
Specifies the gateway node that can be used to run the prefetch operation on a fileset.

"getnThreads": *Integer value*
Specifies the number of threads that is to be used for the prefetch operation. The valid values are 1-255 and the default is 4.

"force":*True | False*
Enables the option to forcefully fetch data from the home cluster during the migration process.

"enableFailedFileList":*True | False*
Enables the creation of a list of files that failed during the prefetch operation at the gateway node.

"metaDataOnly":*True | False*
Enables prefetching of only the metadata and not the actual data. This option is useful in migration scenarios. It must be combined with the **listFile** option.

"listFile': *File name and location*
Specifies the name and location of the file that comprises the list of files that must be pre-populated. The files are listed one file per line.

"dirListFile":*File name and location*
Specifies the individual directories under the AFM fileset that must be prefetched.

"homeFileSystemPath":*File path location*
Specifies the location of the home cluster.

"evict"
The parameters for starting the deallocation of data blocks.

"safeLimit": *Number of files to be evicted*
The list of files that you want to evict. The soft limit creates the benchmark for the target quota limit, which cannot exceed this limit. Each line contains one file. All files must have fully qualified domain names (FQDN).

"fileName": "*Name of the file*"
The fully qualified name of the file that needs to be evicted.

"logFile":"*File name and location*"
Specifies the file, which stores the eviction log. By default, an eviction log file is not generated.

"listfile":"*File name and location*"
Specifies the name and location of the file that comprises the list of files that must be evicted. The files are listed on separate lines and must have fully qualified domain names (FQDN). You need not specify the file system quotas. If the list includes files, which have special characters in their names then you can use a policy to generate the **listfile** parameter.

"filePath":"*File path location*"
Specifies the fully qualified domain name (FQDN) of the file that needs to be evicted.

"calledSetterCount": "*Called setter count*",
Called setter count.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
```

```
 "jobs": [
    {
       "result":"",
         {
           "commands":"String",
           "progress":"String,
           "exitCode":"Exit code",
           "stderr":"Error",
           "stdout":"String",
         },
       "request":" ",
         {
           "type":"{GET | POST | PUT | DELETE}",
           "url":"URL",
           "data":"",
         }
       "jobId":"ID",
       "submitted":"Time",
       "completed":"Time",
       "status":"Job status",
       }
 ],
}
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero denotes success and any value other than zero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr*.

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example creates and links a fileset `gpfs_cache_iw_0` in file system `gpfs0`.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json'
--header 'accept:application/json' -d '{ \
  "failback": {
    "action": "start",
    "failoverTime": "10"
  },
  "failoverToSecondary": {
    "restore": true
  },
  "failover": {
    "newTarget": "gpfs:///afmr/remote_homeFS/gpfs_home_iw_0",
    "targetOnly": true
  },
  "convertToPrimary": {
    "afmtarget": "nfs://9.11.102.209/gpfs/fs0/nfs-ganesha1",
    "inband": true,
    "secondarySnapName": "Snap1",
    "noCheckMetadata": true,
    "rpo": "true"
  },
  "convertToSecondary": {
    "primaryid": "88888888",
    "force": true
  },
  "changeSecondary": {
    "newTarget": "nfs://9.11.102.209/gpfs/fs0/nfs-ganesha1",
    "inband": true,
    "targetOnly": true
  },
  "replacePrimary": {},
  "failbackToPrimary": {
    "action": "start",
    "force": true
  },
  "applyUpdates": {},
  "prefetch": {
    "policy": true,
    "gatewayNode": "Node2",
    "force": true,
    "enableFailedFileList": true,
    "metaDataOnly": true,
    "listFile": "/tmp/file1",
    "dirListFile": "/tmp/file1",
    "homeFileSystemPath": "/gpfs/remotefs1",
    "threadsCount": 4
  },
  "evict": {
    "filter": true,
    "safeLimit": 1,
    "fileName": " /gpfs/fs1/ro2/file10M_1",
    "logFile": "/gpfs/fs1/log",
    "listfile": "/tmp/file-list",
    "filePath": " /gpfs/fs1/ro2/file10M_1"
  },
  "calledSetterCount": 0
} ' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/gpfs_cache_iw_0/afmctl'
```

Response data:

```
{
  "jobs": [
    {
      "jobId": 1000000000002,
      "status": "COMPLETED",
      "submitted": "2021-06-07 10:51:46,963",
      "completed": "2021-06-07 10:51:47,568",
      "runtime": 605,
      "request": {
        "data": {
```

```
          "newTarget": "gpfs:///afmr/remote_homeFS/gpfs_home_iw_0",
          "targetOnly": true
        },
        "type": "POST",
        "url": "/scalemgmt/v2/filesystems/gpfs0/filesets/gpfs_cache_iw_0/afmctl"
      },
       "result": {
        "progress": [],
        "commands": [
          "mmafmctl 'cacheFS' failover -j 'gpfs_cache_iw_0' --new-target 'gpfs:///afmr/
remote_homeFS/gpfs_home_iw_0' --target-only "
        ],
        "stdout": [
          "mmafmctl: Performing failover to gpfs:///afmr/remote_homeFS/gpfs_home_iw_0",
          "Fileset gpfs_cache_iw_0 changed.",
          "info: mmafmctl: Performing failover to gpfs:///afmr/remote_homeFS/
gpfs_home_iw_0\nFileset gpfs_cache_iw_0 changed.\n"
        ],
        "stderr": [],
        "exitCode": 0
      },
      "pids": []
    }
  ],
  "status": {
    "code": 200,
    "message": "The request finished successfully."
  }
}
```

“mmafmlocal command” on page 78
Provides a list of cached files and file statistics such as inode number, allocated blocks, and so on.

“mmafmconfig command” on page 45
Can be used to manage home caching behavior and mapping of gateways and home NFS exported servers.

“mmafmctl command” on page 61
This command is for various operations and reporting information on all filesets. It is recommended to read the *IBM Spectrum Scale: Administration Guide* AFM and AFM Disaster Recovery chapters along with this manual for detailed description of the functions.

Filesystems/{filesystemName}/filesets/{filesetName}/cos/directory: POST

Creates a directory that is related to a corresponding bucket.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `Filesystems/`*{filesystemName}*`/filesets/`*{filesetName}*`/cos/directory` request creates a directory that is related to a corresponding bucket. For more information about the fields in the data structures that are returned, see "mmafmcosaccess command" on page 48.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/
{filesystemName}/filesets/{filesetName}/cos/directory
```

where

`filesystems/`*{filesystemName}*`/filesets/`*{filesetName}*`/cos/directory`
Specifies the fileset in which the directory needs to be created.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 80. List of parameters

Parameter name	Description and applicable keywords	Required/optional
filesetName	The fileset name. This is the path of the fileset.	Required.
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
  "dir": "Directory name",
  "bucket": "Name of the bucket",
  "endpoint": "URL of the object store",
  "accessKey": "Access key",
```

```
  "secretKey": "Secret key"
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"dir": "*Directory name*"
Name of the directory name or full path inside a file system, where you need to link the fileset. If you skip this parameter, the fileset name is used as the directory name.

"bucket": "*Name of the bucket*"
Name of the bucket that is related to your fileset. You can skip this parameter if your fileset has the same name as your bucket.

"endpoint": "*URL of the object store*"
URL to your object store. Use server's hostname or IP or name of the map.

"accessKey": "*Access key*"
Access key for your bucket. Use together with Secret key.

"secretKey": "*Secret key*"
Secret key for your bucket. Use together with Access key.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*'
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success. A nonzero value denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example creates a directory that is related to a corresponding bucket.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json' -d '{ \
   "dir": "mydir", \
   "bucket": "mybucket", \
   "endpoint": "http://s3store.com:8080", \
   "accessKey": "ACCESS_KEY", \
   "secretKey": "SECRET_KEY" \
 }' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/mfset1/cos/directory'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000014,
    "status" : "COMPLETED",
    "submitted" : "2020-09-29 16:42:39,680",
    "completed" : "2020-09-29 16:42:39,957",
    "runtime" : 277,
    "request" : {
      "data" : {
        "accessKey" : "****",
        "bucket" : "mybucket",
        "dir" : "mydir",
        "endpoint" : "http://s3store.com:8080",
        "secretKey" : "****"
      },
```

```
      "type" : "POST",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/filesets/mfset1/cos/directory"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmafmcosaccess 'gpfs0' 'mfset1' 'mydir' set --bucket 'mybucket' --
endpoint 'http://s3store.com:8080'  --akey ****  --skey **** " ],
      "stdout" : [ ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Related reference

“mmafmcosaccess command” on page 48
Maps a directory in an AFM to cloud object storage fileset to the bucket on a cloud object storage.

Filesystems/{filesystemName}/filesets/{filesetName}/cos/download: POST

Downloads files from object store.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `Filesystems/`*{filesystemName}*`/filesets/`*{filesetName}*`/cos/download` request downloads a directory from object store. For more information about the fields in the data structures that are returned, see “mmafmcosaccess command” on page 48.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/
{filesystemName}/filesets/{filesetName}/cos/download
```

where

`filesystems/`*{filesystemName}*`/filesets/`*{filesetName}*`/cos/download`
Specifies the action to be performed.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 81. List of parameters

Parameter name	Description and applicable keywords	Required/optional
filesetName	The fileset name. This is the path of the fileset.	Required.
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
  "path": "Path",
  "objectList": "List of files",
  "all": "true | false",
  "useMetadata": "true | false",
  "useData": "true | false",
  "useNoSubdir": "true | false",
```

```
  "prefix": "Prefix",
  "uid": "User ID",
  "gid": "Group ID",
  "permission": "Access permissions"
 }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"path": "*Path*"
Path to a dedicated directory in a fileset. You can skip this parameter to use default link path of your fileset.

"objectList": "*List of files*"
List of files to be downloaded from Cloud Object Storage.

"all": "*true | false*"
Whether to download all files from Cloud Object Storage.

"useMetadata": "*true | false*"
Whether to download only metadata.

"useData": "*true | false*"
Whether to download data.

"useNoSubdir": "*true | false*"
Whether to create a subdirectory, if '/' is in object name.

"prefix": "*Prefix*"
Prefix of object names to download.

"uid": "*User ID*"
User ID of the fileset owner.

"gid": "*Group ID*"
Group ID of the fileset owner.

"permission": "*Access permissions*"
Access permission in octal format.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
        ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success. A nonzero value denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to download files from object store.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json' -d '{ \
   "path": "/fs1/myfset/mydirectory", \
   "objectList": "[file1, file2]", \
   "all": true, \
   "useMetadata": false, \
   "useData": true, \
   "useNoSubdir": true, \
   "prefix": "hpt", \
   "uid": "1050", \
   "gid": "1050", \
   "permission": "0770" \
 }' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/mfset1/cos/download
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.


```
{
  "jobs" : [ {
    "jobId" : 1000000000015,
    "status" : "FAILED",
    "submitted" : "2020-09-29 16:52:23,236",
    "completed" : "2020-09-29 16:52:25,515",
    "runtime" : 2279,
    "request" : {
      "data" : {
        "all" : true
      },
      "type" : "POST",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/filesets/mfset1/cos/download"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmafmcosctl 'gpfs0' 'mfset1' '/mnt/gpfs0/mfset1' download --all " ],
      "stdout" : [ ],
      "stderr" : [ "EFSSG0053C Failed to execute command:     Queued\t    Failed\t
TotalData\n           \t          \t  (approx in Bytes)\n         20\t         0\t
1153433600\n\nNot all Objects are queued for Download.\n\nmmafmcosctl: Fileset fileset1 is
not an AFM fileset.\nAFM: tspcacheu failed, rc 2 code 0\nmmafmcosctl: Command failed. Examine
previous error messages to determine cause.\n." ],
      "exitCode" : 8
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```


Related reference

“mmafmcosaccess command” on page 48
Maps a directory in an AFM to cloud object storage fileset to the bucket on a cloud object storage.

Filesystems/{filesystemName}/filesets/{filesetName}/cos/evict: POST

Evicts files from object store.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `Filesystems/`*{filesystemName}*`/filesets/`*{filesetName}*`/cos/evict` request evicts files from object store. For more information about the fields in the data structures that are returned, see "mmafmcosaccess command" on page 48.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/
{filesystemName}/filesets/{filesetName}/cos/evict
```

where

`filesystems/`*{filesystemName}*`/filesets/`*{filesetName}*`/cos/evict`
Specifies the action to be performed.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 82. List of parameters

Parameter name	Description and applicable keywords	Required/optional
filesetName	The fileset name. This is the path of the fileset.	Required.
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
  "path": "Path",
  "objectList": "List of files",
  "all": "true | false",
  "useMetadata": "true | false",
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"path": "*Path*"
Path to a dedicated directory in a fileset. You can skip this parameter to use default link path of your fileset.

"objectList": "*List of files*"
List of files to be evicted from COS.

"all": "*true | false*"
Whether to evict all files from COS.

"useMetadata": "*true | false*"
Whether to evict only metadata.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success. A nonzero value denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to evict files from object store.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json' -d '{ \
   "path": "/fs1/myfset/mydirectory", \
   "objectList": "[file1, file2]", \
   "all": true, \
   "useMetadata": false \
 }' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/mfset1/cos/evict
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000016,
    "status" : "FAILED",
    "submitted" : "2020-09-29 20:38:44,349",
    "completed" : "2020-09-29 20:38:45,203",
    "runtime" : 854,
    "request" : {
      "data" : {
        "all" : true,
        "path" : "cosdir1",
        "useMetadata" : false
      },
      "type" : "POST",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/filesets/mfset1/cos/evict"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmafmcosctl 'gpfs0' 'mfset1' 'cosdir1' evict --all   " ],
      "stdout" : [ ],
      "stderr" : [ "EFSSG0053C Failed to execute command:\nmmafmctl: Fileset mfset1 is not
an AFM fileset.\nmmafmctl: Command failed. Examine previous error messages to determine
cause.\nmmafmcosctl: Evict cmd: \"/usr/lpp/mmfs/bin/mmafmctl gpfs0 evict -j mfset1 --list-
```

```
file /var/mmfs/tmp/evict-1601404724\" failed with error 1\nmmafmcosctl: Command failed. Examine
previous error messages to determine cause.\n." ],
      "exitCode" : 8
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Related reference

“mmafmcosaccess command” on page 48

Maps a directory in an AFM to cloud object storage fileset to the bucket on a cloud object storage.

Filesystems/{filesystemName}/filesets/{filesetName}/cos/upload: POST

Uploads files to object store.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `Filesystems/`*{filesystemName}*`/filesets/`*{filesetName}*`/cos/upload` request uploads files to object store. For more information about the fields in the data structures that are returned, see "mmafmcosaccess command" on page 48.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/
{filesystemName}/filesets/{filesetName}/cos/upload
```

where

`filesystems/`*{filesystemName}*`/filesets/`*{filesetName}*`/cos/upload`
Specifies the action to be performed.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 83. List of parameters

Parameter name	Description and applicable keywords	Required/optional
filesetName	The fileset name. This is the path of the fileset.	Required.
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
  "path": "Path",
  "objectList": "List of files",
  "all": "true | false",
 }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"path": "*Path*"
Path to a dedicated directory in a fileset. You can skip this parameter to use default link path of your fileset.

"objectList": "*List of files*"
List of files to be uploaded to Cloud Object Storage.

"all": "*true | false*"
Whether to upload all files to Cloud Object Storage.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success. A nonzero value denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to upload files to object store.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json' -d '{ \
   "path": "/fs1/myfset/mydirectory", \
   "objectList": "[file1, file2]", \
   "all": true, \
 }' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/mfset1/cos/upload
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000017,
    "status" : "FAILED",
    "submitted" : "2020-09-29 20:45:50,393",
    "completed" : "2020-09-29 20:45:50,899",
    "runtime" : 506,
    "request" : {
      "data" : {
        "all" : true,
        "path" : "cosdir1"
      },
      "type" : "POST",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/filesets/fileset1/cos/upload"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmafmcosctl 'gpfs0' 'fileset1' 'cosdir1' upload --all " ],
      "stdout" : [ ],
      "stderr" : [ "EFSSG0053C Failed to execute command:\nmmafmcosctl: Fileset fileset1
is not an AFM fileset.\nmmafmcosctl: cosdir1 is not a valid path for the fileset
fileset1\nmmafmcosctl: Command failed. Examine previous error messages to determine
cause.\n." ],
      "exitCode" : 8
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
```

```
    "message" : "The request finished successfully."
  }
}
```

Related reference

“mmafmcosaccess command” on page 48
Maps a directory in an AFM to cloud object storage fileset to the bucket on a cloud object storage.

Filesystems/{filesystemName}/filesets/{filesetName}/directory/{path}: POST

Creates a directory inside a fileset.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `filesystems/{filesystemName}/filesets/{filesetName}/directory/{path}` request creates directory inside a fileset.

Request URL

```
https://management API host:port/scalemgmt/v2/filesystems/filesystemName/filesets/filesetName/
directory/path
```

where:

`filesystems/filesystemName`
Specifies the name of the file system to which the fileset belong. Required.

`filesets/filesetName`
Specifies the name of the fileset to which the directory belongs. Required.

`directory/path`
Specifies the directory to be created. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 84. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name.	Required.
filesetName		Required.
path	The directory path relative to the fileset path.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
"user": "User name",
```

```
"uid": "User ID",
"group": "Group name",
"gid": "Group ID"
    }
```

"user": "*User name*"
The name of the owning user.

"uid": "*User ID*"
The ID of the owning user.

"group": "*Group name*"
The name of the owning user group.

"gid": "*Group ID*"
The ID of the owning user group.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
   ],
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to create a directory inside the fileset `fs1`.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
-d '{
   "user": "testuser55",
   "uid": 1234,
   "group": "mygroup",
   "gid": 4711
}' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/fs1/directory/mydir'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000005,
    "status" : "RUNNING",
    "submitted" : "2017-03-14 15:59:34,180",
    "completed" : "N/A",
    "request" : {
      "data" : {
         "user": "testuser55",
         "uid": 1234,
         "group": "mygroup",
         "gid": 4711
      },
      "type" : "POST",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/filesets/fs1/directory/mydir"
    },
    "result" : { }
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing"
  }
}
```

Filesystems/{filesystemName}/filesets/{filesetName}/directory/{path}: DELETE

Removes a directory from a fileset.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `DELETE filesystems/{filesystemName}/filesets/{filesetName}/directory/{path}` request removes a directory from a fileset.

Request URL

```
https://management API host:port/scalemgmt/v2/filesystems/filesystemName/filesets/filesetName/
directory/path
```

where:

filesystems/filesystemName
Specifies the name of the file system to which the fileset belongs. Required.

filesets/filesetName
Specifies the fileset. Required.

directory/path
The path of the directory to be removed.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 85. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name.	Required.
filesetName	The fileset name.	Required.
path	The directory path relative to the fileset path.	Required.

Request data

None.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
```

```
 },
 "jobs": [
    {
       "result":"",
         {
           "commands":"String",
           "progress":"String,
           "exitCode":"Exit code",
           "stderr":"Error",
           "stdout":"String",
         },
       "request":" ",
         {
           "type":"{GET | POST | PUT | DELETE}",
           "url":"URL",
           "data":"",
         }
       "jobId":"ID",
       "submitted":"Time",
       "completed":Time",
       "status":"Job status",
       }
 ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to remove a directory from the fileset myFset1.

Request data:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/directory/myDir1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000006,
    "status" : "RUNNING",
    "submitted" : "2017-03-14 16:05:30,960",
    "completed" : "N/A",
    "request" : {
            "type" : "DELETE",
            "url" : "/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/directory/myDir1"
    },
    "result" : { }
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing"
  }
}
```

Filesystems/{filesystemName}/filesets/{filesetName}/directoryCopy/{sourcePath}: PUT

Copies a directory on a fileset.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `filesystems/{filesystemName}/filesets/{filesetName}/directoryCopy/{sourcePath}` request copies a directory on a particular fileset. For more information about the fields in the data structures that are returned, see the topics "mmchfileset command" on page 224, "mmcrfileset command" on page 311, and "mmlsfileset command" on page 501.

Request URL

```
https://<IP or host name of API server>:port/scalemgmt/v2/filesystems/{filesystemName}/
directoryCopy/{sourcePath}
```

where:

filesystems/filesystemName/filesets/{filesetName}
Specifies the fileset to which the directory belongs. Required.

directoryCopy
Action to be performed on the directory. Required.

sourcePath
Path of the directory to be copied. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 86. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
filesetName	The fileset name. This is the path of the fileset.	Optional.
sourcePath	Path of the directory to be copied.	Required.

Request data

```
{
"targetFilesystem": "File system name",
"targetFileset": "Fileset name",
```

```
"targetPath": "Directory path",
"nodeclassName": "Name of the node class",
"force": "True | False",
}
```

"targetFilesystem": "*File system name*"
The name of the file system where the directory is located.

"targetFileset": "*Fileset name*"
The name of the fileset where the directory is located. This is optional.

"targetPath": "*Directory path*"
The name of the file system where the directory is located.

"nodeclassName": "*Name of the node class*"
The name of the node class.

"force": "*File system name*"
Specifies whether the **cp** command is used with **--force** option.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
        ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*'
Array of commands that are run in this job.

"progress":"*String*'
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to copy a directory that belongs to the file system `gpfs0` and fileset `fset1`.

Request data:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
-d '{"targetFilesystem": "gpfs0", \
"targetFileset": "fileset1", \
"targetPath": "cosdir10", \
"nodeclassName": "cesNodes", \
"force": true}'  'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/fileset1/
directoryCopy/cosdir1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000019,
    "status" : "COMPLETED",
    "submitted" : "2020-09-29 20:55:05,814",
    "completed" : "2020-09-29 20:55:08,544",
    "runtime" : 2730,
    "request" : {
      "type" : "PUT",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/filesets/fileset1/directoryCopy/cosdir1"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "tscp --source '/mnt/gpfs0/fileset1/cosdir1' --target '/mnt/gpfs0/fileset1/
cosdir10' --nodeclass 'cesNodes' --force " ],
      "stdout" : [ ],
      "stderr" : [ ],
      "exitCode" : 0
    },
```

```
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

See also

- “mmchfileset command” on page 224
- “mmcrfileset command” on page 311
- “mmlsfileset command” on page 501

Filesystems/{filesystemName}/filesets/{filesetName}/link: DELETE

Unlinks a fileset.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `DELETE filesystems/{filesystemName}/filesets/{filesetName}/link` request unlinks an existing fileset, which is part of the specified file system. For more information about the fields in the data structures that are returned, see “mmunlinkfileset command” on page 735.

Request URL

```
https://management API host:port/scalemgmt/v2/filsystems/filesystemName/filesets/filesetName/
link
```

where:

`filesystem/filesystemName`
Specifies the name of the file system to which the fileset belongs. Required.

`filesets/filesetName`
Specifies fileset to be unlinked. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 87. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
filesetName	The fileset name. This is the path of the fileset.	Required.
force	Specifies whether to force the unlink operation. Default value is false.	Optional.

Request data

```
{ "force": "True | False"
    }
```

"force": "*True | False*"
Forces the unlink to take place even though there are open files.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
   ],
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example unlinks the fileset `myFset1`.

Request data:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json' -d '{
  "force": "true"
}' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/link'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000006,
    "status" : "RUNNING",
    "submitted" : "2017-03-14 16:05:30,960",
    "completed" : "N/A",
    "request" : {
      "data" : {
        "force" : false
      },
      "type" : "DELETE",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/link"
    },
    "result" : { }
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing"
  }
}
```

See also

- “mmunlinkfileset command” on page 735

Filesystems/{filesystemName}/filesets/{filesetName}/link: POST

Links an existing fileset.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `filesystems/{filesystemName}/filesets/{filesetName}/link` request links an existing fileset, which is part of the specified file system. For more information about the fields in the data structures that are returned, see “mmlinkfileset command” on page 485.

Request URL

```
https://management API host:port/scalemgmt/v2/filsystems/filesystemName/filesets/filesetName/
link
```

where:

`filesystem/filesystemName`
Specifies the name of the file system to which the fileset belong. Required.

`filesets/filesetName`
Specifies fileset to be linked. Required.

`link`
Specifies action to be performed in the POST call. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 88. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
filesetName	The fileset name. This is the path of the fileset.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{ "path": "Path"
    }
```

"path": "*Path*"
The absolute path of the fileset.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
   ],
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example links the fileset myFset1.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
-d '{
  "path": "/mnt/gpfs0/fset1"
}' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/link'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000005,
    "status" : "RUNNING",
    "submitted" : "2017-03-14 15:59:34,180",
    "completed" : "N/A",
    "request" : {
      "data" : {
        "path" : "/mnt/gpfs0/fset1"
      },
      "type" : "POST",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/link"
    },
    "result" : { }
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing"
  }
}
```

See also

- “mmlinkfileset command” on page 485

Filesystems/{filesystemName}/filesets/{filesetName}/psnaps: POST

Creates an AFM peer snapshot.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `filesystems/{filesystemName}/filesets/{filesetName}/psnaps` request creates an AFM peer snapshot. The peer snapshot function provides a snapshot at home and cache sites separately, ensuring application consistency on both home and cache sides. For more information about the fields in the data structures that are returned, see the topics "mmpsnap command" on page 614, "mmafmctl command" on page 61, and "mmafmconfig command" on page 45.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/{filesystemName}
/filesets/{filesetName}/psnaps
```

where

filesystems/{filesystemName}/filesets/{filesetName}
Specifies the AFM fileset as the target. Required.

psnap
Specifies that a peer snapshot needs to be taken for the AFM fileset.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 89. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
filesetName	The fileset name. This is the path of the fileset.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Optional.

Request data

```
  {
  "comment": "Comment",
  "uid": "Location",
  "rpo": " yes | no",
  "wait": "yes | no"
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"comment":"*Comment*"
Comment about the peer snapshot.

"uId":"*UID*"
A unique identifier for the cache site. If not specified, this defaults to the IBM Spectrum Scale cluster ID.

"rpo":"*yes | no*"
Specifies whether to create a user recovery point objective (RPO) snapshot for a primary fileset. This option cannot be specified with the --comment and --uid options.

"wait":"*yes | no*"
Specifies whether to make the creation of cache and home snapshots a synchronous process.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following API command creates a peer snapshot of the fileset *myFset1*.

Request URL:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
 -d "{
  "comment": "My peer snapshot",
  "uid": "HONGKONG",
  "rpo": "no",
  "wait": "yes""
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/psnaps'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": "[''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": "[''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": "[''EFSSG0740C There are not enough resources available to create
                  a new independent file set.'',  ...]",
        "stdout": "[''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "POST",
        "url": "/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/psnaps",
        "data":  "comment": "My peer snapshot",
               "uid": "HONGKONG",
               "rpo": "no",
              "wait": "yes",
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

“mmpsnap command” on page 614
Creates or deletes identical snapshots on the cache and home clusters, or shows the status of snapshots that have been queued up on the gateway nodes.

“mmafmconfig command” on page 45
Can be used to manage home caching behavior and mapping of gateways and home NFS exported servers.

“mmafmctl command” on page 61
This command is for various operations and reporting information on all filesets. It is recommended to read the *IBM Spectrum Scale: Administration Guide* AFM and AFM Disaster Recovery chapters along with this manual for detailed description of the functions.

Filesystems/{filesystemName}/filesets/{filesetName}/psnaps/{snapshotName}: DELETE

Deletes a specific AFM peer snapshot.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `DELETE filesystems/{filesystemName}/filesets/{filesetName}/psnaps/{snapshotName}` request deletes a specific AFM peer snapshot. For more information about the fields in the data structures that are returned, see the topics "mmpsnap command" on page 614 and "mmdelsnapshot command" on page 383.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/
{filesystemName}/filesets
/{filesetName}/psnaps/{snapshotName}
```

where

filesystems/{filesystemName}/filesets/{filesetName}
Specifies the AFM fileset as the target. Required.

psnap/{snapshotName}
Specifies the peer snapshot to be deleted.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 90. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
filesetName	The fileset name. This is the path of the fileset.	Required.
snapshotName	The snapshot name. This is the path of the snapshot.	Required.

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*'
Array of commands that are run in this job.

"progress":"*String*'
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following API command deletes the peer snapshot *snap1*.

Request URL:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/psnaps/snap1'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": "[''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": "[''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": "[''EFSSG0740C There are not enough resources available to create
                  a new independent file set.'',  ...]",
        "stdout": "[''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "DELETE",
        "url": "/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/psnaps/snap1",
        },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

“mmpsnap command” on page 614

Creates or deletes identical snapshots on the cache and home clusters, or shows the status of snapshots that have been queued up on the gateway nodes.

“mmdelsnapshot command” on page 383
Deletes a GPFS snapshot.

Filesystems/{filesystemName}/filesets/{filesetName}/quotadefaults: GET

Gets a list of default quotas that are defined in the cluster. With no parameters, all quota defaults will be returned. For all numbers returned, the unit in which the number of blocks is displayed is 1K.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET filesystems/filesystemName/filesets/{filesetName}/quotadefaults` request gets information about the default quotas that are defined at the fileset level. For more information, see “mmsetquota command” on page 702 and “mmrepquota command” on page 666

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/filesystemName
/filesets/filesetName/quotadefaults
```

where

filesystems/filesystemName/filesets/filesetName
: The fileset about which you need the default quota information. Required.

quotadefaults
: Specifies that you need to get the default quota details. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 91. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
filesetName	The fileset name. This is the path of the fileset.	Required.
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Request data

No request data.

Response data

```
      {
         "status": {
             "code":ReturnCode,
             "message":"ReturnMessage"
      },
         "paging": {
         "next": "URL",
         "fields": "Fields specified",
         "filter": "Filter used",
         "baseUrl": "Base URL",
         "lastId": "Last ID"
             },
         "quotaDefaults": [
               {
             "clusterId": {
               "clusterID": "string"
                  },
             "deviceName": "string",
             "filesetId": "Fileset ID",
             "filesetName": "Fileset name",
             "quotaType": "Quota type",
             "blockSoftLimit": "Soft limit set for capacity",
             "blockHardLimit": "Hard limit set for capacity",
             "filesSoftLimit": "Soft limit set for number of inodes",
             "filesHardLimit": "Hard limit set for number of inodes",
             "entryType": "DEFAULT_ON",
   },
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging":"*Paging details*"

"next": "*URL*",
URL of the next item.

"fields": "*Fields specified*",
Fields specified in the request.

"filter": "*Filter used*",
Filters used in the request.

"baseUrl": "*Base URL*",
Base URL

"lastId": "*Last ID*",
Last item's ID.

"quotasDefaults":""

"clusterId"

clusterID": "*ID*"
Unique cluster ID.

"deviceName": "*string*",
Device name.

"filesetId": "*Fileset ID*",
The fileset ID for which the default quota is applicable.

"filesetName": "*Fileset name*",
The fileset for which the quota is applicable.

"quotaType":"USR | GRP | FILESET"
The quota type.

"blockSoftLimit": "*Soft limit set for capacity*",
Soft limit set for capacity.

"blockHardLimit": "*Hard limit set for capacity*",
Hard limit set for capacity.

"filesSoftLimit": "*Soft limit set for number of inodes*",
Soft limit set for number of inodes.

"filesHardLimit": "*Hard limit set for number of inodes*",
Hard limit set for number of inodes.

"entryType": "DEFAULT_ON",
Entry type.

Examples

The following example gets quota information for all filesets inside file system `mari`.

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/:all/quotadefaults'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  },
  "quotaDefaults" : [ {
    "deviceName" : "mari",
    "filesetId" : 0,
    "filesetName" : "root",
    "quotaType" : "USR",
    "blockSoftLimit" : 204800,
    "blockHardLimit" : 409600,
    "filesSoftLimit" : 0,
    "filesHardLimit" : 0,
    "entryType" : "DEFAULT_ON"
  }, {
    "deviceName" : "mari",
    "filesetId" : 0,
    "filesetName" : "root",
    "quotaType" : "GRP",
    "blockSoftLimit" : 0,
    "blockHardLimit" : 0,
    "filesSoftLimit" : 0,
    "filesHardLimit" : 0,
    "entryType" : "DEFAULT_ON"
  }, {
    "deviceName" : "mari",
    "filesetId" : 2,
    "filesetName" : "lisa",
    "quotaType" : "USR",
    "blockSoftLimit" : 0,
    "blockHardLimit" : 0,
    "filesSoftLimit" : 0,
    "filesHardLimit" : 0,
    "entryType" : "DEFAULT_ON"
  }, {
```

```
    "deviceName" : "mari",
    "filesetId" : 2,
    "filesetName" : "lisa",
    "quotaType" : "GRP",
    "blockSoftLimit" : 0,
    "blockHardLimit" : 0,
    "filesSoftLimit" : 0,
    "filesHardLimit" : 0,
    "entryType" : "DEFAULT_OFF"
  } ]
}
```

“mmsetquota command” on page 702
Sets quota limits.

“mmrepquota command” on page 666
Displays file system user, group, and fileset quotas.

Filesystems/{filesystemName}/filesets/{filesetName}/defaultquotas: PUT

Enables default quota limits at file system level.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `filesystems/filesystemName/filesets/filesetName/quotadefaults` request enables default quota limits for a file system. For more information about the fields in the data structures that are returned, see the topics "mmsetquota command" on page 702 and "mmrepquota command" on page 666.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/filesystemName
/filesets/filesetName/quotadefaults
```

where

filesystems/filesystemName/filesets/filesetName
Specifies that the you need to enable default quota for the particular fileset. Required.

quotadefaults
Specifies that you need to set the default quota details. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 92. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name.	Required.
filesetName	The fileset name. This is the path of the fileset.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
          "user":"True | False",
         "group":"True | False",
```

```
                "fileset":"True | False",
                "assign":"True | False",
                "reset":"True | False",
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"user":"*True | False*"
Whether to enable or disable default quota for the user.

"group":"*True | False*"
Whether to enable or disable default quota for the group.

"fileset":"*True | False*"
Whether to enable or disable default quota for the fileset.

"assign":"*True | False*"

Whether to assign the quota defaults to enabled user, group or fileset.

"reset":"*True | False*"
Whether to reset the quota defaults from disabled user, group or fileset.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example enables or disables default quota for the file system `gpfs0` and fileset `fs1`.

Use the following request to set the quota defaults:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
"https://localhost:47443/scalemgmt/v2/filesystems/gpfs0/quotadefaults"
-d '{ "user":true, "group": true, "fileset": true }'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000006,
    "status" : "RUNNING",
    "submitted" : "2019-02-20 13:33:56,336",
    "completed" : "N/A",
    "runtime" : 3,
    "request" : {
      "type" : "PUT",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/filesets/fs1/quotadefaults"
    },
    "result" : { },
    "pids" : [ ]
  }, {
    "jobId" : 1000000000007,
    "status" : "RUNNING",
    "submitted" : "2019-02-20 13:33:56,338",
    "completed" : "N/A",
    "runtime" : 1,
    "request" : {
      "type" : "PUT",
```

```
      "url" : "/scalemgmt/v2/filesystems/gpfs0/filesets/fs1/quotadefaults"
    },
    "result" : { },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing."
  }
}
```

Use the `GET filesystems/gpfs0/filesets/fs1/quotadefaults` request to see how the quota defaults are set:

```
# curl -k -u admin:admin001 -X GET -H content-type:application/json
"https://localhost:443/scalemgmt/v2/filesystems/gpfs0/filesets/fs1/quotadefaults"
{
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  },
  "quotaDefaults" : [ {
    "deviceName" : "gpfs0",
    "filesetId" : 2,
    "filesetName" : "fs1",
    "quotaType" : "GRP",
    "blockSoftLimit" : 0,
    "blockHardLimit" : 0,
    "filesSoftLimit" : 0,
    "filesHardLimit" : 0,
    "entryType" : "DEFAULT_ON"
  }, {
    "deviceName" : "gpfs0",
    "filesetId" : 2,
    "filesetName" : "fs1",
    "quotaType" : "USR",
    "blockSoftLimit" : 0,
    "blockHardLimit" : 0,
    "filesSoftLimit" : 0,
    "filesHardLimit" : 0,
    "entryType" : "DEFAULT_ON"
  } ]
}
```

“mmsetquota command” on page 702
Sets quota limits.

“mmrepquota command” on page 666
Displays file system user, group, and fileset quotas.

Filesystems/{filesystemName}/filesets/{filesetName}quotadefaults: POST

Sets or changes default quota limits at file system level.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `filesystems/filesystemName/filesets/{filesetName}quotadefaults` request sets or changes default quota limits for a fileset. For more information about the fields in the data structures that are returned, see the topics "mmsetquota command" on page 702 and "mmrepquota command" on page 666.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/filesystemName
/filesets/filesetName/quotadefaults
```

where

filesystems/filesystemName/filesets/filesetName
Specifies that the you need to set or change default quota for the particular fileset. Required.

quotadefaults
Specifies that you need to set or change the default quota details. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 93. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name.	Required.
filesetName	The fileset name. This is the path of the fileset.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
          " quotaType":"USR | GROUP | FILESET",
          "blockSoftLimit":"Soft limit for capacity",
```

```
                "blockHardLimit":"Hard limit for capacity",
                "filesSoftLimit":"Soft limit for inodes",
                "filesHardLimit":"Hard limit for inodes",
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

" quotaType":"*USR | GROUP | FILESET*"
Specify whether the quota is enabled for user, group, or fileset.

"blockSoftLimit":"*Soft limit for capacity*"
Soft limit set for the capacity usage. Limit can be specified in KiB, MiB, GiB, or TiB. Default is KiB.

"blockHardLimit":"*Hard limit for capacity*"
Hard limit set for the capacity usage. When hard limit is reached, users cannot perform data writes. Limit can be specified in KiB, MiB, GiB, or TiB. Default is KiB.

"filesSoftLimit":"*Soft limit for inodes*"
Soft limit set for inode space. Limit can be specified in KiB, MiB, GiB, or TiB. Default is KiB.

"filesHardLimit":"*Hard limit for inodes*"
Hard limit set for inodes. Limit can be specified in KiB, MiB, GiB, or TiB. Default is KiB.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to set or change default quota for the file system `gpfs0` and fileset `fset1`.

Now, use the following request to set the quota defaults:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
"https://localhost:47443/scalemgmt/v2/filesystems/mari/filesets/fset1/quotadefaults"
-d '{"quotaType": "GRP", "blockSoftLimit": "10M", "blockHardLimit": "1G", "filesSoftLimit":
"100K", "filesHardLimit": "1M"}'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000009,
    "status" : "RUNNING",
    "submitted" : "2019-02-20 14:28:23,747",
    "completed" : "N/A",
    "runtime" : 0,
    "request" : {
      "type" : "POST",
      "url" : "/scalemgmt/v2/filesystems/mari/filesets/fset1/quotadefaults"
    },
    "result" : { },
    "pids" : [ ]
  } ],
```

```
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing."
  }
}{
  "jobs" : [ {
    "jobId" : 1000000000009,
    "status" : "RUNNING",
    "submitted" : "2019-02-20 14:28:23,747",
    "completed" : "N/A",
    "runtime" : 0,
    "request" : {
      "type" : "POST",
      "url" : "/scalemgmt/v2/filesystems/mari/filesets/fset1/quotadefaults"
    },
    "result" : { },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing."
  }
}
```

Use the `GET filesystems/gpfs0/quotadefaults` request to see how the quota defaults are set:

Request: `# curl -k -u admin:admin001 -X GET -H content-type:application/json "https://localhost:443/scalemgmt/v2/filesystems/mari/filesets/fset1/quotadefaults"`

```
{
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  },
  "quotaDefaults" : [ {
    "deviceName" : "mari",
    "filesetId" : 2,
    "filesetName" : "fset1",
    "quotaType" : "USR",
    "blockSoftLimit" : 0,
    "blockHardLimit" : 0,
    "filesSoftLimit" : 0,
    "filesHardLimit" : 0,
    "entryType" : "DEFAULT_ON"
  }, {
    "deviceName" : "mari",
    "filesetId" : 2,
    "filesetName" : "fset1",
    "quotaType" : "GRP",
    "blockSoftLimit" : 12288,
    "blockHardLimit" : 1048576,
    "filesSoftLimit" : 102400,
    "filesHardLimit" : 1048576,
    "entryType" : "DEFAULT_ON"
  } ]
}
```

“mmsetquota command” on page 702
Sets quota limits.

“mmrepquota command” on page 666
Displays file system user, group, and fileset quotas.

Filesystems/{filesystemName}/filesets/{filesetName}/quotas: GET

Gets information about quota set at the fileset level. With no parameters, all quota limits are returned.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET filesystems/filesystemName/filesets/filesetName/quotas` request gets information about quotas set for filesets within a particular file system. For more information about the fields in the data structures that are returned, see the topics "mmsetquota command" on page 702 and "mmrepquota command" on page 666.

The quota definition at the file system level must be *perfileset* to retrieve the quota information using this API call.

Request URL

```
https://<API server>:<port>/scalemgmt/v2/filesystems/FileSystemName/filesets/filesetName/quotas
```

where

filesystems/filesystemName
The file system that contains the fileset. Required.

filesets/filesetName
The fileset about which you need to get the quota information. Required.

quotas
Specifies that you need to get the quota details. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 94. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
filesetName	The fileset name. This is the path of the fileset.	Required.
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.

Table 94. List of request parameters (continued)		
Parameter name	**Description and applicable keywords**	**Required/optional**
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Request data

No request data.

Response data

```
      {
         "status": {
               "code":ReturnCode,
               "message":"ReturnMessage"
          },
         "paging":"{NFSv4} ",
         "quotas": [
                "quotaID":"ID",
                "filesystemName":"File system name",
                "filesetName":"Fileset name",
                "quotaType":"Type",
                "objectName":"Name",
                "objectId":"ID",
                "blockUsage":"Usage",
                "blockQuota":"Soft limit",
                "blockLimit":"Hard limit",
                "blockInDoubt":"Space in doubt",
                "blockGrace":"Grace period",
                "filesUsage":"Number of files in usage",
                "filesQuota":"Soft limit",
                "filesLimit":"Hard limit",
                "filesInDoubt":"Files in doubt",
                "filesGrace":"Grace period",
                "isDefaultQuota":"Default",
          ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging":"*NFSv4*"

"quotas":""

"quotaID":"*ID*"
Internal ID used for paging.

"filesystemName":"*File system name*"
The file system for which the quota is applicable.

"filesetName":"*Fileset name*"
The fileset for which the quota is applicable.

"quotaType":"USR | GRP | FILESET"
The quota type.

"objectName":"*Name*"
Name of the fileset, user, or user group for which the quota is applicable.

"objectId":"*ID*"
Unique identifier of the fileset, user, or user group.

"blockUsage":"*Usage*"
Current capacity quota usage.

"blockQuota":"*Soft limit*"
The soft limit set for the fileset, user, or user group.

"blockLimit":"*Hard limit*"
The hard limit set for the capacity quota usage. A grace period starts when the hard limit is reached.

"blockInDoubt":"*Space in doubt*"
The capacity that is in doubt.

"blockGrace":"*Grace period*"
The grace period set for the capacity quota.

"filesUsage":"*Number of files in usage*"
Number of inodes.

"filesQuota":"*Soft limit*"
The soft limit set for the inode quota.

"filesLimit":"*Hard limit*"
The hard limit set for the inode quota.

"filesInDoubt":"*Files in doubt*"
The files that are in doubt.

"filesGrace":"*Grace period*"
The grace period set for the inode usage.

"isDefaultQuota":"*Default*"
Default quota.

Examples

The following example gets quota information for the fileset `myFset1`, which is part of the file system `gpfs0`.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/quotas'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/quotas?
lastId=1001"
  },
  "quotas": [
    {
      "quotaId": "4711",
      "filesystemName": "gpfs0",
      "filesetName": "myFset1",
      "quotaType": "USR",
      "objectName": "myFset1",
```

```
      "objectId": "128",
      "blockUsage": "0",
      "blockQuota": "2048",
      "blockLimit": "4096",
      "blockInDoubt": "1024",
      "blockGrace": "none",
      "filesUsage": "32",
      "filesQuota": "50",
      "filesLimit": "100",
      "filesInDoubt": "3",
      "filesGrace": "none",
      "isDefaultQuota": false
    }
  ]
}
```

“mmsetquota command” on page 702
Sets quota limits.

“mmrepquota command” on page 666
Displays file system user, group, and fileset quotas.

Filesystems/{filesystemName}/filesets/{filesetName}/quotas: POST

Sets quota limits or default quota limit for a fileset.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `filesystems/filesystemName/filesets/filesetName/quotas` request define quota limits for a fileset, which is part of the specified file system. For more information about the fields in the data structures that are returned, see the topics "mmsetquota command" on page 702 and "mmrepquota command" on page 666.

The quota definition at the file system level must be *perfileset* to successfully run this API command.

Request URL

```
https://<API server>:<port>/scalemgmt/v2/filesystems/FileSystemName/filesets/filesetName/quotas
```

where

filesystems/filesystemName
The file system that contains the fileset. Required.

filesets/filesetName
The fileset for which you need to set the quota information. Required.

quotas
Specifies that you need to set the quota details. Required.

Request headers

```
Accept: application/json
```

Request data

```
{
   "operationType":"Type",
   "quotaType":"Type",
   "blockSoftLimit":"Soft limit",
   "blockHardLimit":"Hard limit",
   "filesSoftLimit":"Soft limit",
   "filesHardLimit":"Hard limit",
   "filesGracePeriod":"Grace period",
   "blockGracePeriod":"Default",
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"operationType":"*Operation type*"
In this case, set quota.

"quotaType":"USR | GRP | FILESET"
The quota type.

"objectName":"*Name*"
Name of the fileset, user, or user group for which the quota is applicable.

"blockSoftLimit":"*Soft limit*"
The soft limit set for the fileset, user, or user group.

"blockHardLimit":"*Hard limit*"
The hard limit set for the capacity quota usage. A grace period starts when the hard limit is reached.

"filesSoftLimit":"*Soft limit*"
The soft limit set for the inode quota.

"filesHardLimit":"*Hard limit*"
The hard limit set for the inode quota.

"filesGracePeriod":"*Grace period*"
The grace period set for the inode usage.

"blockGracePeriod":"*Grace period*"
The grace period set for the capacity quota.

Response data

```
{
     "status": {
       "code":ReturnCode",
       "message":"ReturnMessage"
    },
    jobs: [
       {
          "result":"",
            {
              "commands":"String",
              "progress":"String,
              "exitCode":"Exit code",
              "stderr":"Error",
              "stdout":"String",
            },
          "request":" ",
            {
              "type":"{GET | POST | PUT | DELETE}",
              "url":"URL",
              "data":"",
            }
          "jobId":"ID",
          "submitted":"Time",
          "completed":Time",
          "status":"Job status",
          }
        ],
    }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example sets quota for the fileset `myFset1`, which is part of the file system `gpfs0`:

Request data:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
-d '{
  "operationType": "setQuota",
  "quotaType": "user",
  "objectName": "adam",
  "blockSoftLimit": "1M",
  "blockHardLimit": "2M",
  "filesSoftLimit": "1K",
  "filesHardLimit": "2K",
  "filesGracePeriod": "null",
  "blockGracePeriod": "null"
}' "https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/quotas"
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000008,
    "status" : "RUNNING",
    "submitted" : "2017-03-14 16:07:43,474",
    "completed" : "N/A",
    "request" : {
      "data" : {
        "operationType": "setQuota",
        "quotaType": "user",
        "objectName": "adam",
        "blockSoftLimit": "1M",
        "blockHardLimit": "2M",
        "filesSoftLimit": "1K",
        "filesHardLimit": "2K",
        "filesGracePeriod": "null",
        "blockGracePeriod": "null"
      },
      "type" : "POST",
```

```
      "url" : "/scalemgmt/v2/filesystems/gpfs0/quotas"
    },
    "result" : { }
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing"
  }
}
```

“mmsetquota command” on page 702
Sets quota limits.

“mmrepquota command” on page 666
Displays file system user, group, and fileset quotas.

Filesystems/{filesystemName}/filesets/{filesetName}/snapshotCopy/{snapshotName}: PUT

Copies a snapshot on a fileset.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `filesystems/{filesystemName}/filesets/{filesetName}/snapshotCopy/{snapshotName}` request copies a snapshot on a particular fileset. For more information about the fields in the data structures that are returned, see the topics "mmchfileset command" on page 224, "mmcrfileset command" on page 311, and "mmlsfileset command" on page 501.

Request URL

```
https://<IP or host name of API server>:port/scalemgmt/v2/filesystems/{filesystemName}/
snapshotCopy/{snapshotName}
```

where:

filesystems/filesystemName/filesets/{filesetName}
Specifies the fileset to which the snapshot belongs. Required.

snapshotCopy
Action to be performed on the snapshot. Required.

snapshotName
Name of the snapshot to be copied. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 95. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
filesetName	The fileset name. This is the path of the fileset.	Optional.
snapshotName	Name of the snapshot to be copied.	Required.

Request data

```
{
"targetFilesystem": "File system name",
"targetFileset": "Fileset name",
```

```
"targetPath": "Directory path",
"nodeclassName": "Name of the node class",
"force": "True | False",
}
```

"targetFilesystem": "*File system name*"
The name of the file system where the snapshot is located.

"targetFileset": "*Fileset name*"
The name of the fileset where the snapshot is located. This is optional.

"targetPath": "*Directory path*"
The name of the file system where the snapshot is located.

"nodeclassName": "*Name of the node class*"
The name of the node class.

"force": "*File system name*"
Specifies whether the **cp** command is used with **--force** option.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*'
Array of commands that are run in this job.

"progress":"*String*'
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to copy a snapshot that belongs to the file system `gpfs0` and fileset `fset1`.

Request data:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
-d '{
  "targetFilesystem": "gpfs0",
"targetPath":"mydir",
"nodeclassName": "cesNodes",
"force": true}' 'https://9.155.108.233:1000/scalemgmt/v2/filesystems/fs1/filesets/root/
snapshotCopy/snap2'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000026,
    "status" : "COMPLETED",
    "submitted" : "2020-09-29 21:59:35,983",
    "completed" : "2020-09-29 21:59:43,828",
    "runtime" : 7845,
    "request" : {
      "type" : "PUT",
      "url" : "/scalemgmt/v2/filesystems/fs1/filesets/root/snapshotCopy/snap2"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "tscp --snapshot 'fs1:root:snap2' --target '/mnt/gpfs0/mydir' --nodeclass
'cesNodes' --force " ],
      "stdout" : [ ],
      "stderr" : [ ],
      "exitCode" : 0
    },
```

```
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

See also

- “mmchfileset command” on page 224
- “mmcrfileset command” on page 311
- “mmlsfileset command” on page 501

Filesystems/{filesystemName}/filesets/{filesetName}/snapshotCopy/{snapshotName}/path/{sourcePath}: PUT

Copies a directory from a source path relative to a snapshot, to a target path on a fileset.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `filesystems/{filesystemName}/filesets/{filesetName}/snapshotCopy/{snapshotName}/path/{sourcePath}` request copies a directory from a source path relative to a snapshot, to a target path on a fileset. For more information about the fields in the data structures that are returned, see the topics "mmchfileset command" on page 224, "mmcrfileset command" on page 311, and "mmlsfileset command" on page 501.

Request URL

```
https://<IP or host name of API server>:port/scalemgmt/v2/filesystems/{filesystemName}/
snapshotCopy/{snapshotName}/path/{sourcePath}
```

where:

filesystems/filesystemName/filesets/{filesetName}
Specifies the fileset to which the snapshot belongs. Required.

snapshotCopy
Action to be performed on the snapshot. Required.

snapshotName
Name of the snapshot to be copied. Required.

path/sourcePath
Source path relative to a snapshot. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 96. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
filesetName	The fileset name. This is the path of the fileset.	Optional.
snapshotName	Name of the snapshot to be copied.	Required.

Table 96. List of request parameters (continued)		
Parameter name	**Description and applicable keywords**	**Required/optional**
path/{sourcePath}	Source path relative to a snapshot.	Required.

Request data

```
{
"targetFilesystem": "File system name",
"targetFileset": "Fileset name",
"targetPath": "Directory path",
"nodeclassName": "Name of the node class",
"force": "True | False",
}
```

"targetFilesystem": "*File system name*"
The name of the file system where the snapshot is located.

"targetFileset": "*Fileset name*"
The name of the fileset where the snapshot is located. This is optional.

"targetPath": "*Directory path*"
The name of the file system where the snapshot is located.

"nodeclassName": "*Name of the node class*"
The name of the node class.

"force": "*File system name*"
Specifies whether the **cp** command is used with **--force** option.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to copy a snapshot that belongs to the file system `gpfs0` and fileset `fset1`. Snapshot name is `snap1` and the relative source path is `mydir1`.

Request data:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
-d '{
  "targetFilesystem": "gpfs0", \
"targetPath":"mydir", \
"nodeclassName": "cesNodes", \
"force": true}' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/fs1/filesets/root/
snapshotCopy/snap2/path/dir1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000027,
    "status" : "COMPLETED",
    "submitted" : "2020-09-29 22:05:03,274",
    "completed" : "2020-09-29 22:05:09,542",
    "runtime" : 6268,
    "request" : {
      "type" : "PUT",
      "url" : "/scalemgmt/v2/filesystems/fs1/filesets/root/snapshotCopy/snap2/path/dir1"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "tscp --snapshot 'fs1:root:snap2' --source 'dir1' --target '/mnt/gpfs0/
mydir' --nodeclass 'cesNodes' --force " ],
      "stdout" : [ ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

See also

- “mmchfileset command” on page 224
- “mmcrfileset command” on page 311
- “mmlsfileset command” on page 501

Filesystems/{filesystemName}/filesets/{filesetName}/snapshots: GET

Gets information about snapshots of a specific fileset.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET filesystems/{filesystemName}/filesets/{filesetName}/snapshots` request gets information about snapshots in the specified fileset. For more information about the fields in the data structures that are returned, see the topics "mmcrsnapshot command" on page 340 and "mmlssnapshot command" on page 540.

Request URL

```
https://IP of API server:<port>/scalemgmt/v2/filesystems/filesystemName/filesets/filesetName/
snapshots
```

where

filesystems/filesystemName
Specifies the file system to which the fileset belongs. Required.

filesets/filesetName
Specifies the fileset of which the snapshot is taken. Required.

snapshots
Specifies snapshot as the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 97. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
filesetName	The fileset name. This is the path of the fileset.	Required.
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.

Table 97. List of request parameters (continued)		
Parameter name	**Description and applicable keywords**	**Required/optional**
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Request data

No request data.

Response data

```
{
   "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   },
    "paging":
    {
    "next": "URL"
    },
   snapshots: [
      {
            "snapshotName":"Snapshot"
            "filesystemName":"Device",
            "filesetName":"Fileset",
            "oid":"ID",
            "snapID":"ID",
            "status":"Status"
            "created":"DateTime",
            "quotas":"Quotas",
            "snapType":"Type",
      }
   ]
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"snapshotName":"*Snapshot*"
The snapshot name.

"filesystemName":"*Device*"
The file system that is the target of the snapshot.

"filesetName":"Fileset"
For a fileset snapshot, the fileset that is a target of the snapshot.

"oid":"*ID*"
Internal identifier that is used for paging.

"snapID":"*ID*"
The snapshot ID.

"status":"*Status*"
The snapshot status.

"created":"*DateTime*"
The date and time when the snapshot was created.

"quotas":"*Quotas*"
Any quotas that are applied to the fileset.

"snapType":"*Type*"
The AFM type of the snapshot, including "afm_snap", "afm_recovery", "afm_failover", "afm_rpo", "afm_baserpo", and "Invalid".

Examples

The following example gets information about the snapshots of the fileset *myFset1*, which belongs to the file system gpfs0.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/snapshots'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "snapshots" : [ {
      "filesetName" : "myFset1",
      "filesystemName" : "gpfs0",
      "snapshotName" : "mySnap1"
    }
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully"
  }
}
```

Using the field parameter ":all:" returns entire details of the snapshot as shown in the following example:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/snapshots?
fields=:all:'
```

```
{
  "snapshots" : [ {
      "created" : "2017-03-21 15:52:14,000",
      "filesetName" : "myFset1",
      "filesystemName" : "gpfs0",
      "oid" : 2,
      "quotas" : "",
      "snapID" : 2,
      "snapType" : "",
      "snapshotName" : "mySnap1",
      "status" : "Valid"
    }
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully"
  }
}
```

“mmcrsnapshot command” on page 340

Creates a snapshot of a file system or fileset at a single point in time.

"mmlssnapshot command" on page 540
Displays GPFS snapshot information.

Filesystems/{filesystemName}/filesets/{filesetName}/snapshots: POST

Creates a fileset snapshot.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `filesystems/{filesystemName}/filesets/{filesetName}/snapshots` command creates a snapshot of the specified fileset.

Request URL

```
https://IP of API server:<port>/scalemgmt/v2/filesystems/filesystemName/filesets/fileseName/
snapshots
```

where

`filesystems/filesystemName`
Specifies the file system to which the fileset belongs. Required.

`filesets/filesetName`
Specifies the fileset of which the snapshot is taken. Required.

`snapshots`
Specifies snapshot as the resource of this POST call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 98. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
filesetName	The fileset name. This is the path of the fileset.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
  "snapshotName": ""
}
```

"snapshotName":
Name of the snapshot to be created.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example creates a snapshot *snap2* of the fileset *myFset1* in file system `gpfs0`.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
-d '{ \
   "snapshotName": "snap2" \
 }' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/snapshots'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {}
      "request": {
        "type": "POST",
        "url": "/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/snapshots",
        "data": "{"snapshotName": "snap2"}"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

"mmcrsnapshot command" on page 340
Creates a snapshot of a file system or fileset at a single point in time.

"mmlssnapshot command" on page 540
Displays GPFS snapshot information.

Filesystems/{filesystemName}/filesets/{filesetName}/snapshots/{snapshotName}: DELETE

Deletes a fileset snapshot.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `DELETE filesystems/{filesystemName}/filesets/{filesetName}/snapshots/{snapshotName}` command deletes the specific fileset snapshot. For more information on deleting snapshots, see the topic "mmdelsnapshot command" on page 383.

Request URL

Use this URL to delete a snapshot:

```
https://IP address of API server:<port>/scalemgmt/v2/filesystems/filesystemName/filesets/
filesetName
/snapshots/snapshotName
```

where:

filesystems/filesystemName
Specifies the file system to which the fileset belongs. Required.

filesets/filesetName
Specifies the fileset of which the snapshot is deleted. Required.

snapshots/snapshotName
Specifies the snapshot to be deleted. Required.

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 99. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
filesetName	The fileset name. This is the path of the fileset.	Required.
snapshotName	The snapshot name. This is the path of the snapshot.	Required.

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode",
```

```
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
        ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example deletes the fileset snapshot *snap1* from the file system *gpfs0*.

Request URL:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/snapshots/snap1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {},
      "request": {
        "type": "DELETE",
        "url": "scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/snapshots/snap1",
        "data": "{}"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

“mmcrsnapshot command” on page 340
Creates a snapshot of a file system or fileset at a single point in time.

“mmlssnapshot command” on page 540
Displays GPFS snapshot information.

Filesystems/{filesystemName}/filesets/{filesetName}/snapshots/{snapshotName}: GET

Gets information about a specific fileset snapshot.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET filesystems/filesystemName/filesets/filesetName/snapshots/snapshotName` request gets information about a specific fileset snapshot in a specific file system. For more information about the fields in the data structures that are returned, see the topics "mmcrsnapshot command" on page 340 and "mmlssnapshot command" on page 540.

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/filesystems/filesystemName/filesets/
filesetName
/snapshots/snapshotName
```

where

filesystems/filesystemName
: Specifies the file system to which the fileset belongs. Required.

filesets/{filesetName}
: Specifies the fileset of which the snapshot is taken. Required.

snapshots/snapshotName
: Specifies a particular snapshot as the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 100. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
filesetName	The fileset name. This is the path of the fileset.	Required.
snapshotName	The snapshot name. This is the path of the snapshot.	Required.
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.

Table 100. List of request parameters (continued)		
Parameter name	**Description and applicable keywords**	**Required/optional**
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Request data

No request data.

Response data

```
{
   "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   },
    "paging":
    {
    "next": "URL"
    },
   snapshots: [
      {
            "snapshotName":"Snapshot"
            "filesystemName":"Device",
            "filesetName":"Fileset",
            "oid":"ID",
            "snapID":"ID",
            "status":"Status"
            "created":"DateTime",
            "quotas":"Quotas",
            "snapType":"Type",
      }
   ]
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"snapshotName":"*Snapshot*"
The snapshot name.

"filesystemName":"*Device*"
The file system that is the target of the snapshot.

"filesetName":"Fileset"
For a fileset snapshot, the fileset that is a target of the snapshot.

"oid":"*ID*"
Internal identifier that is used for paging.

"snapID":"*ID*"
The snapshot ID.

"status":"*Status*"
The snapshot status.

"created":"*DateTime*"
The date and time when the snapshot was created.

"quotas":"*Quotas*"
Any quotas that are applied to the fileset.

"snapType":"*Type*"
The AFM type of the snapshot, including "afm_snap", "afm_recovery", "afm_failover", "afm_rpo", "afm_baserpo", and "Invalid".

Examples

The following example gets information about the snapshots *snap1* of the fileset *myFset1*, which belongs to the file system gpfs0.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/snapshots/snap1
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "paging": {
   "next": "https://localhost:443/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/snapshots/
snap1?lastId=1001"
  },
  "snapshots": [
    {
        "snapshotName": "snap1",
        "filesystemName": "gpfs0",
        "filesetName": "myFset1",
        "oid": "123",
        "snapID": "5",
        "status": "Valid",
        "created": "2017-01-09 14.55.37",
        "quotas": "string",
        "snapType": "string"
    }
  ]
}
```

“mmcrsnapshot command” on page 340
Creates a snapshot of a file system or fileset at a single point in time.

“mmlssnapshot command” on page 540
Displays GPFS snapshot information.

Filesystems/{filesystemName}/suspend: PUT

Suspends a file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `filesystems/filesystemName/suspend` request suspends a file system. For more information about the fields in the data structures that are returned, see “mmfsctl command” on page 424.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/FileSystemName/
suspend
```

where

filesystems/*filesystemName*
The file system that is going to be mounted. Required.

suspend
Specifies the action to be performed on the file system. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 101. List of parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.

Request data

None.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
```

```
            "exitCode":"Exit code",
            "stderr":"Error",
            "stdout":"String",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted":"Time",
        "completed":Time",
        "status":"Job status",
        }
  ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":"*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example how to suspend the file system `gpfs0`.

Request URL:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
}' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/suspend'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000002,
    "status" : "RUNNING",
    "submitted" : "2017-03-14 15:50:00,493",
    "completed" : "N/A",
    "request" : {

      "type" : "PUT",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/suspend"
    },
    "result" : { }
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing"
  }
}
```

“mmfsctl command” on page 424
Issues a file system control request.

Filesystems/{filesystemName}/filesets/{filesetName}/symlink/{linkpath}: POST

Create a symlink for the path of the fileset.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `filesystems/{filesystemName}/filesets/{filesetName}/symlink/{linkPath}` request creates a symlink for the path of a fileset.

Request URL

```
https://management API host:port/scalemgmt/v2/filesystems/filesystemName/filesets/filesetName/
symlink/linkPath
```

where:

`filesystem/filesystemName`
Specifies the name of the file system to which the fileset belongs. Required.

`filesets/filesetName`
Specifies the name of the fileset to which the symlink belongs. Required.

`symlink/linkPath`
Specifies the symlink path relative to the fileset path. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 102. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name.	Required.
filesetName	The fileset name.	Required.
linkpath	The symlink path relative to the fileset path.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation. This is the target path of the symlink relative to the file system's mount point or fileset path.	Required.

Request data

```
{
"filesystemName": "File system name",
"filesetName": "Fileset name",
"relativePath": "Relative path"
    }
```

"filesystemName": "*File system name*"
The name of the file system.

"filesetName": "*Fileset name*"
The name of the fileset.

"relativePath": "*Relative path*"
The target path of the symlink relative to the file system's mount point or fileset path.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
   ],
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to create a symlink for the fileset `fs1`.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
-d '{
  "filesystemName": "gpfs0",
   "filesetName": "fset1",
   "relativePath": "mydir"
}' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/fset1/symlink/mydir'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000005,
    "status" : "RUNNING",
    "submitted" : "2017-03-14 15:59:34,180",
    "completed" : "N/A",
    "request" : {
      "data" : {
                "filesystemName": "gpfs0",
                "filesetName": "fset1",
                "relativePath": "mydir"
                },
      "type" : "POST",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/filesets/fs1/symlink/mydir"
    },
    "result" : { }
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing"
  }
}
```

Filesystems/{filesystemName}/filesets/{filesetName}/symlink/{path}: DELETE

Removes a symlink from a fileset.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `DELETE filesystems/{filesystemName}/filesets/{filesetName}/symlink/{path}` request removes a symlink from a fileset.

Request URL

```
https://management API host:port/scalemgmt/v2/filesystems/filesystemName/filesets/filesetName/
symlink/path
```

where:

filesystems/filesystemName
Specifies the name of the file system to which the fileset belongs. Required.

filesets/filesetName
Specifies the name of the fileset to which the symlink belongs. Required.

symlink/path
The symlink path relative to the fileset path

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 103. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name.	Required.
filesetName	The fileset name.	Required.
path	The symlink path relative to the fileset path	Required.

Request data

None.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
```

```
 },
 "jobs": [
    {
       "result":"",
         {
           "commands":"String",
           "progress":"String,
           "exitCode":"Exit code",
           "stderr":"Error",
           "stdout":"String",
         },
       "request":" ",
         {
           "type":"{GET | POST | PUT | DELETE}",
           "url":"URL",
           "data":"",
         }
       "jobId":"ID",
       "submitted":"Time",
       "completed":Time",
       "status":"Job status",
       }
 ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to remove symlink from the fileset myFset1.

Request data:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/symlink/myDir1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000006,
    "status" : "RUNNING",
    "submitted" : "2017-03-14 16:05:30,960",
    "completed" : "N/A",
    "request" : {
      "type" : "DELETE",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/symlink/myDir1"
    "result" : { }
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing"
  }
}
```

Filesystems/{filesystemName}/filesets/{filesetName}/watch: PUT

Enables or disables clustered watch for a fileset.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `filesystems/filesystemName/filesets/{filesetName}/watch` request enables or disables clustered watch for a fileset. For more information about the fields in the data structures that are returned, see “mmwatch command” on page 764.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/
{FileSystemName}/filesets/{filesetName}/watch
```

where

filesystems/*filesystemName*/filesets/*filesetName*
: The fileset for which clustered watch needs to be enabled.

watch
: Specifies the action to be performed on the fileset.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 104. List of parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
filesetName	The fileset name. This is the path of the fileset.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{

  "action": "Enable or disable file system watch",
  "watchfolderConfig":
  {
```

```
    "description": "Description",
    "eventTypes": "Event types",
    "eventHandler": "Event handler",
    "sinkBrokers": "Sink broker",
    "sinkTopic": "Sink topic",
    "sinkAuthConfig": "Sink authentication config"
  },
  "watchId": "Watch ID"
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"action": "*Enable or disable file system watch*"
Whether to enable or disable clustered watch for the fileset. Possible values are `enable | disable`.

"watchfolderConfig":

"description": "*Description*"
Description of the clustered watch.

"eventTypes": "*Event types*"
Types of events that need to be watched.

"eventHandler": "*Event handler*"
Type of event handler. Only Kafkasink is supported as the evens handler.

"sinkBrokers": "*Sink broker*"
Includes a comma-separated list of broker:port pairs for the sink Kafka cluster, which is the external Kafka cluster where the events are sent.

"sinkTopic": "*Sink topic*"
The topic that producers write to in the sink Kafka cluster.

"sinkAuthConfig": "*Sink authentication config*"
The path to the sink authentication configuration file.

"watchId": "*Watch ID*"
Remote cluster name from where the file system must be watched.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
   ],
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":"*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example how to enable the clustered watch for the fileset `fset1`.

Request data:

```
{
  "action": "enable",
  "watchfolderConfig": {
    "description": "description",
    "eventTypes": "IN_ACCESS,IN_ATTRIB",
    "eventHandler": "kafkasink",
    "sinkBrokers": "Broker1:Port,Broker2:Port",
    "sinkTopic": "topic",
    "sinkAuthConfig": "/mnt/gpfs0/sink-auth-config"
  },
  "watchId": "string"
}
```

Corresponding request URL:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
-d '{ \
   "action": "enable", \
   "watchfolderConfig": { \
     "description": "description", \
     "eventTypes": "IN_ACCESS,IN_ATTRIB", \
     "eventHandler": "kafkasink", \
     "sinkBrokers": "Broker1:Port,Broker2:Port", \
     "sinkTopic": "topic", \
     "sinkAuthConfig": "/mnt/gpfs0/sink-auth-config" \
   }, \
   "watchId": "string" \
 }' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filsets/fset1/watch'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": ["''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": ["''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": ["''EFSSG0740C There are not enough resources available to create
                   a new independent file set.'',  ...]",
        "stdout": ["''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "PUT",
        "url": "/scalemgmt/v2/filesystems/gpfs0/filsets/fset1/watch",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

“mmwatch command” on page 764
Administers clustered watch folder watches.

Filesystems/{filesystemName}/mount: PUT

Performs mount operation of a specific file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `filesystems/filesystemName/mount` request mounts a file system. For more information about the fields in the data structures that are returned, see “mmmount command” on page 545.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/FileSystemName/
mount
```

where

filesystems/*filesystemName*
: The file system that is going to be mounted. Required.

mount
: Specifies the action to be performed on the file system. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 105. List of parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
            "nodes":"{Node name} ",
          "mountOptions": "Mount options",
           ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"nodes":"*Node name*"
The node on which the file system must be mounted.

"mountOption":"*Mount options*"
Additional mount options, which override the defaults and are not persistent.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
   ],
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":"*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example how to mount the file system `gpfs0`.

Request data:

```
{
  "nodes": "testnode-11",
  "mountOptions": "atime=yes;relatime=no"
}
```

Corresponding request URL:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
-d '{ "nodes": "testnode-11",
  "mountOptions": "atime=yes;relatime=no"
}
}' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/mount'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000002,
    "status" : "RUNNING",
    "submitted" : "2017-03-14 15:50:00,493",
    "completed" : "N/A",
    "request" : {
      "data" : {
        "entries" : [
      {
          "type" : "allow",
          "who" : "special:owner@",
          "permissions" : "rwmxDaAnNcCos",
          "flags" : ""
        },
         {
          "type" : "allow",
          "who" : "special:group@",
          "permissions" : "rxancs",
          "flags" : ""
        },
        {
          "type" : "allow",
          "who" : "special:everyone@",
          "permissions" : "rxancs",
          "flags" : ""
        },
        {
          "type" : "allow",
          "who" : "user:scalemgmt",
```

```
          "permissions" : "rxancs",
          "flags" : "fd"
        } ],
        "type" : "NFSv4"
      },
      "type" : "PUT",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/mount"
    },
    "result" : { }
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing"
  }
}
```

“mmmount command” on page 545
Mounts GPFS file systems on one or more nodes in the cluster.

Filesystems/{filesystemName}/owner/{path}: GET

Gets information about owner of a file or directory.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET filesystems/filesystemName/owner/path` request gets information about owner of files or directories within a particular file system.

Request URL

```
https://<IP address of API server>:<port>/scalemgmt/v2/filesystems/filesystemName/owner/path
```

where

filesystems/filesystemName
The file system about which you want to get the information. Required.

owner/path
The path of the file or directory about which you want to get the owner information. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 106. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
path	The file path relative to file system's mount point. The path of the file or directory is specified with forward slashes (/). For example: `mnt/gpfs0/rest01`. The forward slashes in the path are encoded to "%2F" in the request URL.	Required.

Request data

No request data.

Response data

```
{
      "status": {
          "code":ReturnCode,
          "message":"ReturnMessage"
       },
         "owner": [
                "user":"User name",
                "uid":"User ID",
                "group":"Group name",
                "gid":"Group ID",
          ],
 }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"Owner":"*Owner details*"

"user":"*User name*"
Name of the owner.

"uid":"*User ID*"
Unique identifier of the owner.

"group":"*Group name*"
Name of the user group that owns the file or directory.

"gid":"*Group ID*"
Unique identifier of the user group that owns the file or directory.

Examples

The following example gets owner information for the files and directories of the file system `gpfs0`.

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/owner/2Fmnt%2Fgpfso%2Fxaz'
```

Response URL:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."k
  },
  "owner": {
    "user": "testuser55",
    "uid": "1234",
    "group": "mygroup",
    "gid": "4711"
  }
}
```

"mmcrfs command" on page 318

Creates a GPFS file system.

"mmchfs command" on page 232
Changes the attributes of a GPFS file system.

"mmlsfs command" on page 506
Displays file system attributes.

Filesystems/{filesystemName}/owner/{path}: PUT

Sets owner for a file or directory.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `filesystems/filesystemName/owner/path` request sets owner for files or directories within a particular file system.

The file and directory ownership can be controlled by Linux commands such as **`stat`** and **`chown`**. You can use either ID or name of the user or group to define the ownership. If you use both ID and name, the ID takes precedence. Only the user with *DataAccess* role can change the owner of a file or a non-empty directory. Empty directories can be changed by other roles such as Administrator, ProtocolAdmin, StorageAdmin, and SecurityAdmin.

Request URL

```
https://<IP address or hostname of API server>:<port>/scalemgmt/v2/filesystems/FilesystemName/
owner/path
```

where

filesystemName
: Specifies the file system to which the file or directory belongs. Required.

path
: The path of the file or directory for which you want to set the owner. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 107. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
path	The file path relative to file system's mount point. The path of the file or directory is specified with forward slashes (/). For example: `mnt/gpfs0/rest01`. The forward slashes in the path are encoded to "%2F" in the request URL.	Required.

Table 107. List of request parameters (continued)		
Parameter name	**Description and applicable keywords**	**Required/optional**
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
                 "path":"Path",
                "user":"User name",
                "uid":"User ID",
                "group":"Group name",
                "gid":"Group ID",
                "permission":"Permissions"
                "recursive": "yes | no"
}
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"path":"*Path*"
The path of the file or directory.

"user":"*User name*"
Name of the owner.

"uid":"*User ID*"
Unique identifier of the owner.

"group":"*Group name*"
Name of the user group that owns the file or directory.

"gid":"*Group ID*"
Unique identifier of the user group that owns the file or directory.

"permission":"*Access permissions*"
The number of permissions that are set by using the CLI command **chmod**. If nothing is specified, then no action is allowed.

"recursive":"*true | false*"
Specifies whether the owner is defined recursively for the entire file system or for the specific directory that is defined in the endpoint path.

Response data

```
{
      "status": {
          "code":ReturnCode,
          "message":"ReturnMessage"
       },
         "owner": [
               "path":"Path",
                "user":"User name",
                "uid":"User ID",
                "group":"Group name",
                "gid":"Group ID",
                "permissions": Number of Permissions,
                "recursive": true | false
          ],
 }
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"Owner":"*Owner details*"

"path":"*Path*"
The path of the file or directory.

"user":"*User name*"
Name of the owner.

"uid":"*User ID*"
Unique identifier of the owner.

"group":"*Group name*"
Name of the user group that owns the file or directory.

"gid":"*Group ID*"
Unique identifier of the user group that owns the file or directory.

"permissions":"*Number of permissions*"
The number of permissions that are set by using the CLI command **chmod**.

"recursive":*true | false*
Specifies whether owner is defined recursively for the entire file system or for the specific directory that is defined in the endpoint path.

Examples

The following example sets owner for the files and directories of the file system gpfs0.

Request data:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
-d '{
  "path": "/mnt/gpfs0/xaz",
  "user": "testuser55",
  "uid": "1234",
  "group": "mygroup",
  "gid": "4711"
  "permissions": 700
  "recursive" : "true"
}' "https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/owner/%2Fmnt%2Fgpfs0%2Fxaz"
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs": [
    {
      "jobId": 1000000000002,
      "status": "COMPLETED",
      "submitted": "2021-07-08 09:44:35,020",
      "completed": "2021-07-08 10:00:31,022",
      "runtime": 7,
      "request": {
        "data": {
          "gid": 4711,
          "group": "mygroup",
          "permissions": 700,
          "recursive": false,
          "uid": 1234,
          "user": "testuser55"
```

```
        },
        "type": "PUT",
        "url": "/scalemgmt/v2/filesystems/gpfs0/owner/mydir"
      },
      "result": {},
      "pids": []
    }
  ],
  "status": {
    "code": 202,
    "message": "The request finished successfully."
  }
}
```

“mmcrfs command” on page 318
Creates a GPFS file system.

“mmchfs command” on page 232
Changes the attributes of a GPFS file system.

“mmlsfs command” on page 506
Displays file system attributes.

Filesystems/{filesystemName}/policies: GET

Gets the details of the information lifecycle management (ILM) policies applicable to a specific file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET filesystems/{filesystemName}/policies` request gets the details of the ILM policies applicable to a specific file system. For more information about the fields in the data structures that are returned, see “mmapplypolicy command” on page 80 and “mmlspolicy command” on page 526.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/
{filesystemName}/policies
```

where

`filesystems/{filesystemName}/policies`
Specifies that the GET request fetches the policies that are applicable to a specific file system.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 108. List of parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Request data

No request data.

Response data

The following list of attributes are available in the response data:

```
{
"status":
   {
      "code": ReturnCode
      "message": "ReturnMessage",
   },
"paging":
   {
      "next": Next page URL
      "fields": "Fields",
      "filter": Filter
      "baseUrl": "URL",
      "lastID": ID
      ,
   },
{
   "policies": [
      {
         "filesystemName": "File system name"
         "policy": "Policy settings"
       }
}
```

"status":
Return status.

"code": *ReturnCode*,
The HTTP status code that was returned by the request.

"message": "*ReturnMessage*"
The return message.

"paging":
An array of information about the paging information that is used for displaying the details.

"next": "*Next page URL*"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects would be returned by the query.

"fields": "*Fields in the request*"
The fields that are used in the original request.

"filter": "*Filters used in the request*"
The filter that is used in the original request.

"baseUrl": "*URL*"
The URL of the request without any parameters.

"lastId": "*ID*"
The ID of the last element that can be used to retrieve the next elements.

"policies":
An array of information about the ILM policies applicable to a file system.

"filesystemName": "*File system name*"
The file system for which the ILM policies are applicable.

"policy": "*Policy settings*"
Details of the ILM policy applicable to the specified file system.

The return information and the information that the command retrieves are returned in the same way as they are for the other requests. The parameters that are returned are the same as the configuration attributes that are displayed by the `mmaplypolicy` command. For more information, see “mmlspolicy command” on page 526 and “mmapplypolicy command” on page 80.

Examples

The following example gets information about the cluster configuration.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/policies'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

The **policies** array provides information about the ILM policies applicable to the file system *gpfs0*.

```
{
  "status": {
    "code": 200,
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/filesystems/gpfs0/filesets?lastId=10001",
    "fields": "period,restrict,sensorName",
    "filter": "usedInodes>100,maxInodes>1024",
    "baseUrl": "/scalemgmt/v2/perfmon/sensor/config",
    "lastId": 10001
  },
  "policies": [
    {
      "filesystemName": "gpfs0",
      "policy": "RULE 'placement' SET POOL 'system'"
    }
  ]
}
```

“mmapplypolicy command” on page 80
Deletes files, migrates files between storage pools, or does file compression or decompression in a file system as directed by policy rules.

“mmlspolicy command” on page 526
Displays policy information.

Filesystems/{filesystemName}/policies: PUT

Sets a policy for a file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `filesystems/{filesystemName}/policies` request applies an Information Lifecycle Management (ILM) policy to a specific file system. For more information about the fields in the data structures that are returned, see “mmapplypolicy command” on page 80.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/
{filesystemName}/policies
```

where

filesystems/{filesystemName}/policies
Specifies the policy for a specific file system as the target of the operation. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 109. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
  {
  "policy": "RULE 'placement' SET POOL 'system'"
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"policy":"*ILM policy*"
Details of the policy to be applied to the file system.

Response data

```
{
     "status": {
       "code":ReturnCode",
       "message":"ReturnMessage"
    },
    jobs: [
        {
           "result":"",
             {
               "commands":"String",
               "progress":"String,
               "exitCode":"Exit code",
               "stderr":"Error",
               "stdout":"String",
             },
           "request":" ",
             {
               "type":"{GET | PUT | PUT | DELETE}",
               "url":"URL",
               "data":"",
             }
           "jobId":"ID",
           "submitted":"Time",
           "completed":Time",
           "status":"Job status",
           }
         ],
    }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*'
Array of commands that are run in this job.

"progress":"*String*'
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | PUT | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following API command creates a threshold rule *rule1*.

Request URL:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
-d '{
    "policy": "RULE 'placement' SET POOL 'system'"
 }'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/policies'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": ["'mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": ["''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": ["'EFSSG0740C There are not enough resources available to create
                  a new independent file set.'',  ...]",
        "stdout": ["'EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "PUT",
        "url": "/scalemgmt/v2/filesystems/gpfs0/policies",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

Related reference

“mmapplypolicy command” on page 80
Deletes files, migrates files between storage pools, or does file compression or decompression in a file system as directed by policy rules.

Filesystems/{filesystemName}/quotadefaults: GET

Gets the details of the default quotas that are defined at the file system level.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET filesystems/filesystemName/quotadefaults` request gets information about the default quotas that are defined at the file system level. For more information, see "mmsetquota command" on page 702 and "mmrepquota command" on page 666

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/filesystemName/
quotadefaults
```

where

filesystems/filesystemName
The file system about which you need the default quota information. Required.

quotadefaults
Specifies that you need to get the default quota details. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 110. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Request data

No request data.

Response data

```
      {
         "status": {
             "code":ReturnCode,
             "message":"ReturnMessage"
      },
         "paging": {
         "next": "URL",
         "fields": "Fields specified",
         "filter": "Filter used",
         "baseUrl": "Base URL",
         "lastId": "Last ID"
             },
         "quotaDefaults": [
               {
             "clusterId": {
               "clusterID": "string"
                  },
             "deviceName": "string",
             "filesetName": "Fileset name",
             "quotaType": "Quota type",
             "blockSoftLimit": "Soft limit set for capacity",
             "blockHardLimit": "Hard limit set for capacity",
             "filesSoftLimit": "Soft limit set for number of inodes",
             "filesHardLimit": "Hard limit set for number of inodes",
             "entryType": "DEFAULT_ON",
             ],
   },
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging":"*Paging details*"

"next": "*URL*",
URL of the next item.

"fields": "*Fields specified*",
Fields specified in the request.

"filter": "*Filter used*",
Filters used in the request.

"baseUrl": "*Base URL*",
Base URL

"lastId": "*Last ID*",
Last item's ID.

"quotasDefaults":""

"clusterId"

clusterID": "*ID*"
Unique cluster ID.

"deviceName": "*string*",
Device name.

"filesetId": "*Fileset ID*",
The fileset ID for which the default quota is applicable.

"filesetName": "*Fileset name*",
The fileset for which the quota is applicable.

"quotaType":"USR | GRP | FILESET"
The quota type.

"blockSoftLimit": "*Soft limit set for capacity*",
Soft limit set for capacity.

"blockHardLimit": "*Hard limit set for capacity*",
Hard limit set for capacity.

"filesSoftLimit": "*Soft limit set for number of inodes*",
Soft limit set for number of inodes.

"filesHardLimit": "*Hard limit set for number of inodes*",
Hard limit set for number of inodes.

"entryType": "DEFAULT_ON",
Entry type.

Examples

The following example gets quota information for all filesets inside file system `mari`.

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/:all/quotadefaults'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  },
  "quotaDefaults" : [ {
    "deviceName" : "mari",
    "filesetId" : 0,
    "filesetName" : "root",
    "quotaType" : "USR",
    "blockSoftLimit" : 204800,
    "blockHardLimit" : 409600,
    "filesSoftLimit" : 0,
    "filesHardLimit" : 0,
    "entryType" : "DEFAULT_ON"
  }, {
    "deviceName" : "mari",
    "filesetId" : 0,
    "filesetName" : "root",
    "quotaType" : "GRP",
    "blockSoftLimit" : 0,
    "blockHardLimit" : 0,
    "filesSoftLimit" : 0,
    "filesHardLimit" : 0,
    "entryType" : "DEFAULT_ON"
  }, {
    "deviceName" : "mari",
    "filesetId" : 2,
    "filesetName" : "lisa",
    "quotaType" : "USR",
    "blockSoftLimit" : 0,
    "blockHardLimit" : 0,
    "filesSoftLimit" : 0,
    "filesHardLimit" : 0,
    "entryType" : "DEFAULT_ON"
  }, {
    "deviceName" : "mari",
    "filesetId" : 2,
    "filesetName" : "lisa",
```

```
    "quotaType" : "GRP",
    "blockSoftLimit" : 0,
    "blockHardLimit" : 0,
    "filesSoftLimit" : 0,
    "filesHardLimit" : 0,
    "entryType" : "DEFAULT_OFF"
  } ]
}
```

“mmsetquota command” on page 702
Sets quota limits.

“mmrepquota command” on page 666
Displays file system user, group, and fileset quotas.

Filesystems/{filesystemName}/quotadefaults: PUT

Enables default quota limits at file system level.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `filesystems/filesystemName/quotadefaults` request enables default quota limits for a file system. For more information about the fields in the data structures that are returned, see the topics "mmsetquota command" on page 702 and "mmrepquota command" on page 666.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/filesystemName/
quotadefaults
```

where

filesystems/filesystemName
Specifies that the you need to enable default quota for the particular file system. Required.

quotadefaults
Specifies that you need to enable the default quota details. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 111. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
  "user":"True | False
  "group":"True | False",
  "fileset":"True | False",
  "assign":"True | False",
  "reset":"True | False",
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"user":"*True | False*"
Whether to enable or disable default quota for the user.

"group":"*True | False*"
Whether to enable or disable default quota for the group.

"fileset":"*True | False*"
Whether to enable or disable default quota for the fileset.

"assign":"*True | False*"

Whether to assign the quota defaults to enabled user, group or fileset.

"reset":"*True | False*"
Whether to reset the quota defaults from disabled user, group or fileset.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example enables or disables default quota for the file system gpfs0.

Use the following request to see the current default settings:

```
# curl -k -u admin:admin001 -X GET -H content-type:application/json
"https://localhost:443/scalemgmt/v2/filesystems/gpfs0/quotadefaults"
{
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  },
  "quotaDefaults" : [ {
    "deviceName" : "gpfs0",
    "filesetId" : 0,
    "quotaType" : "FILESET",
    "blockSoftLimit" : 0,
    "blockHardLimit" : 0,
    "filesSoftLimit" : 0,
    "filesHardLimit" : 0,
    "entryType" : "DEFAULT_OFF"
  }, {
    "deviceName" : "gpfs0",
    "filesetId" : 0,
    "quotaType" : "USR",
    "blockSoftLimit" : 12288,
    "blockHardLimit" : 1048576,
    "filesSoftLimit" : 102400,
    "filesHardLimit" : 1048576,
    "entryType" : "DEFAULT_OFF"
  }, {
    "deviceName" : "gpfs0",
    "filesetId" : 0,
    "quotaType" : "GRP",
    "blockSoftLimit" : 0,
    "blockHardLimit" : 0,
    "filesSoftLimit" : 0,
    "filesHardLimit" : 0,
    "entryType" : "DEFAULT_OFF"
  } ]
}
```

Now, use the following request to set the quota defaults:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
"https://localhost:47443/scalemgmt/v2/filesystems/gpfs0/quotadefaults"
-d '{ "user":true, "group": true, "fileset": true }'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000001,
    "status" : "RUNNING",
    "submitted" : "2019-02-19 15:18:34,738",
    "completed" : "N/A",
    "runtime" : 5,
    "request" : {
      "type" : "PUT",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/quotadefaults"
    },
    "result" : { },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing."
  }
}
```

Use the `GET filesystems/gpfs0/quotadefaults` request to see how the quota defaults are set:

```
# curl -k -u admin:admin001 -X GET -H content-type:application/json
"https://localhost:443/scalemgmt/v2/filesystems/gpfs0/quotadefaults"
{
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  },
  "quotaDefaults" : [ {
    "deviceName" : "gpfs0",
    "filesetId" : 0,
    "quotaType" : "USR",
    "blockSoftLimit" : 12288,
    "blockHardLimit" : 1048576,
    "filesSoftLimit" : 102400,
    "filesHardLimit" : 1048576,
    "entryType" : "DEFAULT_ON"
  }, {
    "deviceName" : "gpfs0",
    "filesetId" : 0,
    "quotaType" : "GRP",
    "blockSoftLimit" : 0,
    "blockHardLimit" : 0,
    "filesSoftLimit" : 0,
    "filesHardLimit" : 0,
    "entryType" : "DEFAULT_ON"
  }, {
    "deviceName" : "gpfs0",
    "filesetId" : 0,
    "quotaType" : "FILESET",
    "blockSoftLimit" : 0,
    "blockHardLimit" : 0,
    "filesSoftLimit" : 0,
    "filesHardLimit" : 0,
    "entryType" : "DEFAULT_ON"
  } ]
}
```

“mmsetquota command” on page 702
Sets quota limits.

“mmrepquota command” on page 666
Displays file system user, group, and fileset quotas.

Filesystems/{filesystemName}/quotadefaults: POST

Sets or changes default quota limits at file system level.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `filesystems/filesystemName/quotadefaults` request sets or changes default quota limits for a file system. For more information about the fields in the data structures that are returned, see the topics “mmsetquota command” on page 702 and “mmrepquota command” on page 666.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/filesystemName/
quotadefaults
```

where

filesystems/filesystemName
Specifies that the you need to set or change default quota for the particular file system. Required.

quotadefaults
Specifies that you need to set or change the default quota details. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 112. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
      {
                " quotaType":"USR | GROUP | FILESET",
                "blockSoftLimit":"Soft limit for capacity",
                "blockHardLimit":"Hard limit for capacity",
                "filesSoftLimit":"Soft limit for inodes",
                "filesHardLimit":"Hard limit for inodes",
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

" quotaType":"*USR | GROUP | FILESET*"
Specify whether the quota is enabled for user, group, or fileset.

"blockSoftLimit":"*Soft limit for capacity*"
Soft limit set for the capacity usage. Limit can be specified in KiB, MiB, GiB, or TiB. Default is KiB.

"blockHardLimit":"*Hard limit for capacity*"
Hard limit set for the capacity usage. When hard limit is reached, users cannot perform data writes. Limit can be specified in KiB, MiB, GiB, or TiB. Default is KiB.

"filesSoftLimit":"*Soft limit for inodes*"
Soft limit set for inode space. Limit can be specified in KiB, MiB, GiB, or TiB. Default is KiB.

"filesHardLimit":"*Hard limit for inodes*"
Hard limit set for inodes. Limit can be specified in KiB, MiB, GiB, or TiB. Default is KiB.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to set or change default quota for the file system `gpfs0`.

Now, use the following request to set the quota defaults:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
"https://localhost:47443/scalemgmt/v2/filesystems/gpfs0/quotadefalts"
-d '{  "quotaType": "USR",  "blockSoftLimit": "10M",  "blockHardLimit": "1G",
"filesSoftLimit": "100K",  "filesHardLimit": "1M" }'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000008,
    "status" : "RUNNING",
    "submitted" : "2019-02-20 14:25:05,280",
    "completed" : "N/A",
    "runtime" : 3,
    "request" : {
      "type" : "POST",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/quotadefaults"
    },
    "result" : { },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing."
  }
}
```

Use the `GET filesystems/gpfs0/quotadefaults` request to see how the quota defaults are set:

```
{
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  },
  "quotaDefaults" : [ {
    "deviceName" : "gpfs0",
    "filesetId" : 0,
    "quotaType" : "USR",
    "blockSoftLimit" : 12288,
    "blockHardLimit" : 1048576,
    "filesSoftLimit" : 102400,
    "filesHardLimit" : 1048576,
    "entryType" : "DEFAULT_ON"
  }, {
    "deviceName" : "gpfs0",
    "filesetId" : 0,
    "quotaType" : "GRP",
    "blockSoftLimit" : 0,
    "blockHardLimit" : 0,
    "filesSoftLimit" : 0,
    "filesHardLimit" : 0,
    "entryType" : "DEFAULT_ON"
  }, {
    "deviceName" : "gpfs0",
    "filesetId" : 0,
    "quotaType" : "FILESET",
    "blockSoftLimit" : 0,
    "blockHardLimit" : 0,
    "filesSoftLimit" : 0,
    "filesHardLimit" : 0,
    "entryType" : "DEFAULT_OFF"
  } ]
}
```

“mmsetquota command” on page 702
Sets quota limits.

“mmrepquota command” on page 666
Displays file system user, group, and fileset quotas.

Filesystems/{filesystemName}/quotagracedefaults: GET

Gets the details of the default grace period set for the quota limits that are defined at the file system level. The grace period starts when soft limit that is defined for capacity or inode usage is reached. You can set grace periods for both capacity and inode limits. User can continue the data writes until the grace time expires or hard limit is reached.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET filesystems/filesystemName/quotagracedefaults` request gets information about the default grace period set for the quotas limits that are defined at the file system level. For more information, see "mmsetquota command" on page 702 and "mmrepquota command" on page 666

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/filesystemName/
quotagracedefaults
```

where

filesystems/filesystemName
The file system about which you need the default quota grace period information. Required.

quotagracedefaults
Specifies that you need to get the default quota grace period details. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 113. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Request data

No request data.

Response data

```
      {
         "status": {
             "code":ReturnCode,
             "message":"ReturnMessage"
      },
         "paging": {
         "next": "URL",
         "fields": "Fields specified",
         "filter": "Filter used",
         "baseUrl": "Base URL",
         "lastId": "Last ID"
             },
         "quotaGraceDefaults": [ {
             "deviceName": "File system name",
             "quotaType": "USR | GRP | FILESET",
             "blockGracePeriod": "Grace period set for capacity usage",
             "filesGracePeriod": "Grace period set for inode usage",
            ],
   },
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging":"*Paging details*"

"next": "*URL*",
URL of the next item.

"fields": "*Fields specified*",
Fields specified in the request.

"filter": "*Filter used*",
Filters used in the request.

"baseUrl": "*Base URL*",
Base URL

"lastId": "*Last ID*",
Last item's ID.

"quotasDefaults":""

"deviceName": "*File system name*",
File system name.

"quotaType":"USR | GRP | FILESET"
The quota type.

"blockGracePeriod": "*Grace period set for capacity usage*",
Grace period set for capacity usage.

"filesGracePeriod": "*Grace period set for inode usage*",
Grace period set for inode usage.

Examples

The following example gets grace periods set for all file systems that are created in the system.

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
"https://localhost:47443/scalemgmt/v2/filesystems/:all:/quotagracedefaults"
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  },
  "quotaGraceDefaults" : [ {
    "deviceName" : "gpfs1",
    "quotaType" : "GRP",
    "blockGracePeriod" : 604800,
    "filesGracePeriod" : 604800
  }, {
    "deviceName" : "gpfs1",
    "quotaType" : "FILESET",
    "blockGracePeriod" : 604800,
    "filesGracePeriod" : 604800
  }, {
    "deviceName" : "gpfs1",
    "quotaType" : "USR",
    "blockGracePeriod" : 1000,
    "filesGracePeriod" : 864000
  }, {
    "deviceName" : "mari",
    "quotaType" : "USR",
    "blockGracePeriod" : 604800,
    "filesGracePeriod" : 604800
  }, {
    "deviceName" : "mari",
    "quotaType" : "FILESET",
    "blockGracePeriod" : 604800,
    "filesGracePeriod" : 604800
  }, {
    "deviceName" : "mari",
    "quotaType" : "GRP",
    "blockGracePeriod" : 36000,
    "filesGracePeriod" : 3600
  } ]
}
```

“mmsetquota command” on page 702
Sets quota limits.

“mmrepquota command” on page 666
Displays file system user, group, and fileset quotas.

Filesystems/{filesystemName}/quotagracedefaults: POST

Sets or changes default grace period set for quota limits at the file system level. The grace period starts when soft limit that is defined for capacity or inode usage is reached. You can set grace periods for both capacity and inode limits. User can continue the data writes until the grace time expires or hard limit is reached.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `filesystems/filesystemName/quotagracedefaults` request sets or changes grace period for quota limits for a file system. For more information about the fields in the data structures that are returned, see the topics "mmsetquota command" on page 702 and "mmrepquota command" on page 666.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/filesystemName/
quotagracedefaults
```

where

filesystems/filesystemName
: Specifies that the you need to set or change grace periods for quota limits for the particular file system. Required.

quotagracedefaults
: Specifies that you need to set or change the default quota details. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 114. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
          " grace":"USR | GRP | FILESET",
          "blockGracePeriod":"Grace period for capacity limits",
```

```
                "filesGracePeriod":"Grace period for inode space limits",
        }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

" grace":"*USR | GROUP | FILESET*"
Specify whether grace period is set for user, group, or fileset.

"blockGracePeriod":"*Grace period for capacity limits*"
Grace period set for the capacity usage. Grace period can be set in *seconds, minutes, hour*s, and *days*. Default unit is *seconds*.

""filesGracePeriod":"*Grace period for inode space limits*"
Grace period set for the inode space usage. Grace period can be set in *seconds, minutes, hours*, and *days*. Default unit is *seconds*.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to set or change grace period for the file system `gpfs0`.

Now, use the following request to set the quota defaults:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
"https://localhost:47443/scalemgmt/v2/filesystems/gpfs0/quotagracedefaults"
-d '{ "grace" : "GRP" , "blockGracePeriod" : "10hours", "filesGracePeriod" : "1hours" }'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000008,
    "status" : "RUNNING",
    "submitted" : "2019-02-20 14:25:05,280",
    "completed" : "N/A",
    "runtime" : 3,
    "request" : {
      "type" : "POST",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/quotadefaults"
    },
    "result" : { },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing."
  }
}
```

Use the `GET filesystems/gpfs0/quotagracedefaults` request to see how the quota defaults are set:

```
# curl -k -u admin:admin001 -XGET -H content-type:application/json
"https://localhost:47443/scalemgmt/v2/filesystems/gpfs0/quotagracedefaults"
{
  "status" : {
```

```
    "code" : 200,
    "message" : "The request finished successfully."
  },
  "quotaGraceDefaults" : [ {
    "deviceName" : "gpfs0",
    "quotaType" : "USR",
    "blockGracePeriod" : 604800,
    "filesGracePeriod" : 604800
  }, {
    "deviceName" : "gpfs0",
    "quotaType" : "FILESET",
    "blockGracePeriod" : 604800,
    "filesGracePeriod" : 604800
  }, {
    "deviceName" : "gpfs0",
    "quotaType" : "GRP",
    "blockGracePeriod" : 36000,
    "filesGracePeriod" : 3600
  } ]
}
```

“mmsetquota command” on page 702
Sets quota limits.

“mmrepquota command” on page 666
Displays file system user, group, and fileset quotas.

Filesystems/{filesystemName}/quotamanagement: PUT

Enables default quota management at the file system level.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `filesystems/filesystemName/quotamanagement` request enables quota management for a file system. For more information about the fields in the data structures that are returned, see the topics "mmsetquota command" on page 702 and "mmrepquota command" on page 666.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/filesystemName/
quotamanagement
```

where

filesystems/filesystemName
Specifies that the you need to enable quota management for the particular file system. Required.

quotamanagement
Specifies that you need to enable quota management. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 115. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
      {
                "quota":"filesystem | fileset | disabled",
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"quota":"*filesystem | fileset | disabled*"
filesystem enables quota management at file system level. *fileset* enables quota management at fileset level. *disabled* disables quota management for the specified file system.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
        ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example enables quota management at fileset level.

```
# curl -k -u admin:admin001 -X PUT -H content-type:application/json
"https://localhost:47443/scalemgmt/v2/filesystems/gpfs0/quotamanagement" -d '{  "quota":
"fileset" }'
{
  "jobs" : [ {
    "jobId" : 1000000000001,
    "status" : "RUNNING",
    "submitted" : "2019-02-26 13:52:03,743",
    "completed" : "N/A",
    "runtime" : 3,
    "request" : {
      "type" : "PUT",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/quotamanagement"
    },
    "result" : { },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing."
  }
}
```

Use the following request to see the quota management settings:

```
# curl -k -u admin:admin001 -X GET -H content-type:application/json
"https://localhost:443/scalemgmt/v2/filesystems/gpfs0?fields=quota"
{
  "filesystems" : [ {
    "name" : "gpfs0",
    "quota" : {
      "defaultQuotasEnabled" : "user;group",
      "filesetdfEnabled" : false,
      "perfilesetQuotas" : true,
      "quotasAccountingEnabled" : "user;group;fileset",
      "quotasEnforced" : "user;group;fileset"
    }
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

“mmsetquota command” on page 702
Sets quota limits.

“mmrepquota command” on page 666
Displays file system user, group, and fileset quotas.

Filesystems/{filesystemName}/quotas: GET

Gets information about quota set at the file system level. With no parameters, all quota limits are returned.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET filesystems/filesystemName/quotas` request gets information about quotas set at file system level. “mmsetquota command” on page 702 and “mmrepquota command” on page 666

The *perfileset* quota must be disabled to display the user and group quotas. The fileset quota is also available in the response data.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/filesystemName/
quotas
```

where

filesystems/filesystemName
The file system about which you need the information. Required.

quotas
Specifies that you need to get the quota details. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 116. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Request data

No request data.

Response data

```
      {
         "status": {
             "code":ReturnCode,
             "message":"ReturnMessage"
      },
         "paging":"{NFSv4} ",
         "quotas": [
                "quotaID":"ID",
                "filesystemName":"File system name",
                "filesetName":"Fileset name",
                "quotaType":"Type",
                "objectName":"Name",
                "objectId":"ID",
                "blockUsage":"Usage",
                "blockQuota":"Soft limit",
                "blockLimit":"Hard limit",
                "blockInDoubt":"Space in doubt",
                "blockGrace":"Grace period",
                "filesUsage":"Number of files in usage",
                "filesQuota":"Soft limit",
                "filesLimit":"Hard limit",
                "filesInDoubt":"Files in doubt",
                "filesGrace":"Grace period",
                "isDefaultQuota":"Default",
            ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging":"*NFSv4*"

"quotas":""

"quotaID":"*ID*"
Internal ID used for paging.

"filesystemName":"*File system name*"
The file system for which the quota is applicable.

"filesetName":"*Fileset name*"
The fileset for which the quota is applicable.

"quotaType":"USR | GRP | FILESET"
The quota type.

"objectName":"*Name*"
Name of the fileset, user, or user group for which the quota is applicable.

"objectId":"*ID*"
Unique identifier of the fileset, user, or user group.

"blockUsage":"*Usage*"
Current capacity quota usage.

"blockQuota":"*Soft limit*"
The soft limit set for the fileset, user, or user group.

"blockLimit":"*Hard limit*"
The hard limit set for the capacity quota usage. A grace period starts when the hard limit is reached.

"blockInDoubt":"*Space in doubt*"
The capacity that is in doubt.

"blockGrace":"*Grace period*"
The grace period set for the capacity quota.

"filesUsage":"*Number of files in usage*"
Number of inodes.

"filesQuota":"*Soft limit*"
The soft limit set for the inode quota.

"filesLimit":"*Hard limit*"
The hard limit set for the inode quota.

"filesInDoubt":"*Files in doubt*"
The files that are in doubt.

"filesGrace":"*Grace period*"
The grace period set for the inode usage.

"isDefaultQuota":"*Default*"
Default quota.

Examples

The following example gets quota information for the file system `gpfs0`.

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/quotas'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/filesystems/gpfs0/quotas?lastId=1001"
  },
  "quotas": [
    {
      "quotaId": "4711",
      "filesystemName": "gpfs0",
      "filesetName": "myFset1",
      "quotaType": "USR",
      "objectName": "myFset1",
      "objectId": "128",
      "blockUsage": "0",
      "blockQuota": "2048",
      "blockLimit": "4096",
      "blockInDoubt": "1024",
      "blockGrace": "none",
      "filesUsage": "32",
      "filesQuota": "50",
      "filesLimit": "100",
      "filesInDoubt": "3",
      "filesGrace": "none",
      "isDefaultQuota": false
    }
  ]
}
```

"mmsetquota command" on page 702

Sets quota limits.

"mmrepquota command" on page 666
Displays file system user, group, and fileset quotas.

Filesystems/{filesystemName}/quotas: POST

Sets quota limits or default quota limit at file system level.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `filesystems/filesystemName/quotas` request define quota limits for a file system. For more information about the fields in the data structures that are returned, see the topics "mmsetquota command" on page 702 and "mmrepquota command" on page 666.

The *perfileset* quota must be disabled to successfully complete this API command.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/filesystemName/
quotas
```

where

filesystems/filesystemName
Specifies that the you need to set quota for the particular file system. Required.

quotas
Specifies that you need to set the quota details. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 117. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
           "operationType":"Type",
         "quotaType":"Type",
         "blockSoftLimit":"Soft limit",
         "blockHardLimit":"Hard limit",
```

```
                "filesSoftLimit":"Soft limit",
                "filesHardLimit":"Hard limit",
                "filesGracePeriod":"Grace period",
                "blockGracePeriod":"Default",
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"operationType":"*Operation type*"
In this case, set quota.

"quotaType":"USR | GRP | FILESET"
The quota type.

"objectName":"*Name*"
Name of the fileset, user, or user group for which the quota is applicable.

"blockSoftLimit":"*Soft limit*"
The soft limit set for the fileset, user, or user group.

"blockHardLimit":"*Hard limit*"
The hard limit set for the capacity quota usage. A grace period starts when the hard limit is reached.

"filesSoftLimit":"*Soft limit*"
The soft limit set for the inode quota.

"filesHardLimit":"*Hard limit*"
The hard limit set for the inode quota.

"filesGracePeriod":"*Grace period*"
The grace period set for the inode usage.

"blockGracePeriod":"*Grace period*"
The grace period set for the capacity quota.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example sets quota for the file system `gpfs0`.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
-d '{
  "operationType": "setQuota",
  "quotaType": "user",
  "objectName": "adam",
  "blockSoftLimit": "1M",
  "blockHardLimit": "2M",
  "filesSoftLimit": "1K",
  "filesHardLimit": "2K",
  "filesGracePeriod": "null",
  "blockGracePeriod": "null"
}' "https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/quotas"
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {},
      "request": {
        "type": "POST",
        "url": "/scalemgmt/v2/filesystems/gpfs0/quotas",
        "data": "{
           "operationType": "setQuota",
           "quotaType": "user",
           "objectName": "adam",
           "blockSoftLimit": "1M",
           "blockHardLimit": "2M",
           "filesSoftLimit": "1K",
           "filesHardLimit": "2K",
           "filesGracePeriod": "null",
           "blockGracePeriod": "null""}"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

“mmsetquota command” on page 702
Sets quota limits.

“mmrepquota command” on page 666
Displays file system user, group, and fileset quotas.

Filesystems/{filesystemName}/resume: PUT

Resumes a file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `filesystems/filesystemName/resume` request resumes a file system operation. For more information about the fields in the data structures that are returned, see “mmfsctl command” on page 424.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/FileSystemName/
resume
```

where

filesystems/*filesystemName*
: The file system that is going to be mounted. Required.

resume
: Specifies the action to be performed on the file system. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 118. List of parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.

Request data

None.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
```

```
            "exitCode":"Exit code",
            "stderr":"Error",
            "stdout":"String",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted":"Time",
        "completed":Time",
        "status":"Job status",
        }
  ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":"*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example how to resume the file system `gpfs0`.

Request URL:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
}' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/resume'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000002,
    "status" : "RUNNING",
    "submitted" : "2017-03-14 15:50:00,493",
    "completed" : "N/A",
    "request" : {

      "type" : "PUT",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/resume"
    },
    "result" : { }
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing"
  }
}
```

“mmfsctl command” on page 424
Issues a file system control request.

Filesystems/{filesystemName}/snapshotCopy/{snapshotName}: PUT

Copies a snapshot on a file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `filesystems/{filesystemName}/snapshotCopy/{snapshotName}` request copies a directory from a source path relative to a snapshot, to a target path on a file system. For more information about the fields in the data structures that are returned, see the topics “mmcrfs command” on page 318, “mmchfs command” on page 232, and “mmlsfs command” on page 506.

Request URL

```
https://<IP or host name of API server>:port/scalemgmt/v2/filesystems/{filesystemName}/
snapshotCopy/{snapshotName}
```

where:

filesystems/filesystemName
Specifies the file system to which the snapshot belongs. Required.

snapshotCopy
Action to be performed on the snapshot. Required.

snapshotName
Name of the snapshot to be copied. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 119. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
snapshotName	Name of the snapshot to be copied.	Required.

Request data

```
{
"targetFilesystem": "File system name",
"targetFileset": "Fileset name",
"targetPath": "Directory path",
"nodeclassName": "Name of the node class",
```

```
"force": "True | False",
}
```

"targetFilesystem": "*File system name*"
The name of the file system where the snapshot is located.

"targetFileset": "*Fileset name*"
The name of the fileset where the snapshot is located. This is optional.

"targetPath": "*Directory path*"
The name of the file system where the snapshot is located.

"nodeclassName": "*Name of the node class*"
The name of the node class.

"force": "*File system name*"
Specifies whether the **cp** command is used with **--force** option.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*'
Array of commands that are run in this job.

"progress":"*String*'
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to copy a snapshot that belongs to the file system `gpfs0`.

Request data:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
-d '{
  "targetFilesystem": "objfs","nodeclassName": "cesNodes", "force": true}' 'https://
198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/snapshotCopy/snap1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000024,
    "status" : "COMPLETED",
    "submitted" : "2020-09-29 21:25:25,513",
    "completed" : "2020-09-29 21:25:41,217",
    "runtime" : 15704,
    "request" : {
      "type" : "PUT",
      "url" : "/scalemgmt/v2/filesystems/fs1/snapshotCopy/snap1"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "tscp --snapshot 'fs1:snap1' --target '/mnt/objfs/dir1' --nodeclass
'cesNodes' --force " ],
      "stdout" : [ ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
```

```
    }
}
```

See also

- “mmcrfs command” on page 318
- “mmchfs command” on page 232
- “mmlsfs command” on page 506

Filesystems/{filesystemName}/snapshotCopy/{snapshotName}/path/{sourcePath}: PUT

Copies a directory from a source path relative to a snapshot, to a target path on a file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `filesystems/{filesystemName}/snapshotCopy/{snapshotName}/path/{sourcePath}` request copies a directory from a source path relative to a snapshot, to a target path on a file system. For more information about the fields in the data structures that are returned, see the topics "mmcrfs command" on page 318, "mmchfs command" on page 232, and "mmlsfs command" on page 506.

Request URL

```
https://<IP or host name of API server>:port/scalemgmt/v2/filesystems/{filesystemName}/
snapshotCopy/{snapshotName}/path/{sourcePath}
```

where:

filesystems/filesystemName.
Specifies the file system to which the snapshot belongs. Required.

snapshotCopy
Action to be performed on the snapshot. Required.

snapshotName
Name of the snapshot to be copied. Required.

path/sourcePath
Source path relative to a snapshot. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 120. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
snapshotName	Name of the snapshot to be copied.	Required.
path/{sourcePath}	Source path relative to a snapshot.	Required.

Request data

```
{
"targetFilesystem": "File system name",
"targetFileset": "Fileset name",
"targetPath": "Directory path",
"nodeclassName": "Name of the node class",
"force": "True | False",
}
```

"targetFilesystem": "*File system name*"
The name of the file system where the snapshot is located.

"targetFileset": "*Fileset name*"
The name of the fileset where the snapshot is located. This is optional.

"targetPath": "*Directory path*"
The name of the file system where the snapshot is located.

"nodeclassName": "*Name of the node class*"
The name of the node class.

"force": "*File system name*"
Specifies whether the **cp** command is used with **--force** option.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to copy a snapshot that belongs to the file system `gpfs0`. Snapshot name is `snap1` and the relative source path is `mydir1`.

Request data:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
-d '{
  "targetFilesystem": "gpfs0", \
  "targetPath":"mydir", \
  "nodeclassName": "cesNodes", \
  "force": true}' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/fs1/snapshotCopy/snap1/
path/dir1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000025,
    "status" : "COMPLETED",
    "submitted" : "2020-09-29 21:40:31,217",
    "completed" : "2020-09-29 21:40:36,298",
    "runtime" : 5081,
    "request" : {
      "type" : "PUT",
      "url" : "/scalemgmt/v2/filesystems/fs1/snapshotCopy/snap1/path/dir1"
    },
```

```
    "result" : {
      "progress" : [ ],
      "commands" : [ "tscp --snapshot 'fs1:snap1' --source 'dir1' --target '/mnt/gpfs0/mydir'
--nodeclass 'cesNodes' --force " ],
      "stdout" : [ ],
      "stderr" : [ ],
      "exitCode" : 0
    },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

See also

- “mmcrfs command” on page 318
- “mmchfs command” on page 232
- “mmlsfs command” on page 506

Filesystems/{filesystemName}/snapshots: GET

Gets information about snapshots of a specific file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET filesystems/{filesystemName}/snapshots` request gets information about snapshots in the specified file system. For more information about the fields in the data structures that are returned, see the topics "mmcrsnapshot command" on page 340 and "mmlssnapshot command" on page 540.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/filesystemName/
snapshots
```

where

filesystems/filesystemName
Specifies the file system of which the snapshot is taken. Required.

snapshots
Specifies snapshot as the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 121. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Request data

No request data.

Response data

```
{
   "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   },
    "paging":
    {
    "next": "URL"
    },
   snapshots: [
      {
            "snapshotName":"Snapshot"
            "filesystemName":"Device",
            "filesetName":"Fileset",
            "oid":"ID",
            "snapID":"ID",
            "status":"Status"
            "created":"DateTime",
            "quotas":"Quotas",
            "snapType":"Type",
      }
   ]
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"snapshotName":"*Snapshot*"
The snapshot name.

"filesystemName":"*Device*"
The file system that is the target of the snapshot.

"filesetName":"Fileset"
For a fileset snapshot, the fileset that is a target of the snapshot.

"oid":"*ID*"
Internal identifier that is used for paging.

"snapID":"*ID*"
The snapshot ID.

"status":"*Status*"
The snapshot status.

"created":"*DateTime*"
The date and time when the snapshot was created.

"quotas":"*Quotas*"
Any quotas that are applied to the fileset.

"snapType":"*Type*"
The AFM type of the snapshot, including "`afm_snap`", "`afm_recovery`", "`afm_failover`", "`afm_rpo`", "`afm_baserpo`", and "`Invalid`".

Examples

The following example gets information about the snapshots of the file system `gpfs0`.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/snapshots'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/filesystems/gpfs0/snapshots?lastId=1001"
  },
  "snapshots": [
    {
        "snapshotName": "snap1",
        "filesystemName": "gpfs0",
        "filesetName": "",
        "oid": "123",
        "snapID": "5",
        "status": "Valid",
        "created": "2017-01-09 14.55.37",
        "quotas": "string",
        "snapType": "string"
    }
  ]
}
```

“mmcrsnapshot command” on page 340
Creates a snapshot of a file system or fileset at a single point in time.

“mmlssnapshot command” on page 540
Displays GPFS snapshot information.

Filesystems/{filesystemName}/snapshots: POST

Creates a file system snapshot.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `filesystems/{filesystemName}/snapshots` command creates a snapshot of the specified file system.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/filesystemName/
snapshots
```

where

`filesystems/filesystemName`
Specifies the file system of which the snapshot needs to be taken. Required.

`snapshots`
Specifies snapshot as the resource of this POST call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 122. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
  "snapshotName": ""
}
```

"snapshotName":
Name of the snapshot to be created.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example creates a snapshot *snap2* of the file system `gpfs0`.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
-d '{ \
   "snapshotName": "snap2"
 }' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/snapshots'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {},
      "request": {
        "type": "POST",
        "url": "/scalemgmt/v2/filesystems/gpfs0/snapshots",
        "data": "{"snapshotName": "snap2"}"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

"mmcrsnapshot command" on page 340
Creates a snapshot of a file system or fileset at a single point in time.

"mmlssnapshot command" on page 540
Displays GPFS snapshot information.

Filesystems/{filesystemName}/snapshots/{snapshotName}: DELETE

Deletes a file system snapshot.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `DELETE filesystems/{filesystemName}/snapshots/{snapshotName}` command deletes the specified file system snapshot. For more information on deleting snapshots, see "mmdelsnapshot command" on page 383.

Request URL

Use this URL to delete a global snapshot:

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/filesystemName
/snapshots/snapshotName
```

where:

filesystems/filesystemName
Specifies the file system of which the snapshot is deleted. Required.

snapshots/snapshotName
Specifies snapshot to be deleted. Required.

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 123. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
snapshotName	The snapshot name. This is the path of the snapshot.	Required.

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
```

```
            "exitCode":"Exit code",
            "stderr":"Error",
            "stdout":"String",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted":"Time",
        "completed":Time",
        "status":"Job status",
        }
      ],
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example deletes the fileset snapshot *snap1* from the file system *gpfs0*.

Request URL:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/snapshots/snap1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {},
      "request": {
        "type": "DELETE",
        "url": "/scalemgmt/v2/filesystems/gpfs0/snapshots/snap1",
        },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

“mmcrsnapshot command” on page 340
Creates a snapshot of a file system or fileset at a single point in time.

“mmlssnapshot command” on page 540
Displays GPFS snapshot information.

Filesystems/{filesystemName}/snapshots/{snapshotName}: GET

Gets information about a specific file system snapshot.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET filesystems/filesystemName/snapshots/snapshotName` request gets information about a specific file system snapshot. For more information about the fields in the data structures that are returned, see the topics "mmcrsnapshot command" on page 340 and "mmlssnapshot command" on page 540.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/filesystemName
/snapshots/snapshotName
```

where

filesystems/filesystemName
Specifies the file system of which the snapshot is taken. Required.

snapshots/snapshotName
Specifies a particular snapshot as the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 124. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
snapshotName	The snapshot name. This is the path of the snapshot.	Required.
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.

Request data

No request data.

Response data

```
{
   "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   },
    "paging":
    {
    "next": "URL"
    },
   snapshots: [
      {
            "snapshotName":"Snapshot"
            "filesystemName":"Device",
            "filesetName":"Fileset",
            "oid":"ID",
            "snapID":"ID",
            "status":"Status"
            "created":"DateTime",
            "quotas":"Quotas",
            "snapType":"Type",
      }
   ]
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"snapshotName":"*Snapshot*"
The snapshot name.

"filesystemName":"*Device*"
The file system that is the target of the snapshot.

"filesetName":"Fileset"
For a fileset snapshot, the fileset that is a target of the snapshot.

"oid":"*ID*"
Internal identifier that is used for paging.

"snapID":"*ID*"
The snapshot ID.

"status":"*Status*"
The snapshot status.

"created":"*DateTime*"
The date and time when the snapshot was created.

"quotas":"*Quotas*"
Any quotas that are applied to the fileset.

"snapType":"*Type*"
The AFM type of the snapshot, including "`afm_snap`", "`afm_recovery`", "`afm_failover`", "`afm_rpo`", "`afm_baserpo`", and "`Invalid`".

Examples

The following example gets information about the snapshots *snap1* of the file system `gpfs0`.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/snapshots/snap1
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/filesystems/gpfs0/filesets/myFset1/snapshots/
snap1?lastId=1001"
  },
  "snapshots": [
    {
        "snapshotName": "snap1",
        "filesystemName": "gpfs0",
        "filesetName": "myFset1",
        "oid": "123",
        "snapID": "5",
        "status": "Valid",
        "created": "2017-01-09 14.55.37",
        "quotas": "string",
        "snapType": "string"
    }
  ]
}
```

“mmcrsnapshot command” on page 340
Creates a snapshot of a file system or fileset at a single point in time.

“mmlssnapshot command” on page 540
Displays GPFS snapshot information.

Filesystems/{filesystemName}/pools/{poolName}: GET

Gets the information about a particular storage pool.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET filesystems/{filesystemName}/pools/{poolName}` request gets information about the specified storage pool in a file system. For more information about the fields in the data structures that are returned, see “mmlspool command” on page 528.

Request URL

```
https://<IP or host name of API server>:<port>/scalemgmt/v2/filsystems/filesystemName/pools/
poolName
```

where:

filesystem/filesystemName
Specifies the name of the file system to which the storage pools belong. Required.

pools/poolName
Specifies the storage pool about which you need information. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 125. List of parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
PoolName	The storage pool name.	Required.
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.

Request data

No request data.

Response data

```
{
  "status" : {
    "code" : The ReturnCode,
    "message" : "The ReturnMessage."
  },
```

```
  "storagePool" : [ {
    "storagePoolName" : "Name of storage pool",
    "filesystemName" : "Name of file system",
    "totalDataInKB" : Size,
    "freeDataInKB" : Size,
    "totalMetaInKB" : Size,
    "freeMetaInKB" : Size
  } ]
}
}
```

"status":
Return status.

"code":"*ReturnCode*"
The return code.

"message": "*ReturnMessage*",
The return message.

"storagePool":
The array of information listing the storage pool details.

"storagePoolName": "*Name*",
The name of the storage pool for which information is displayed.

"filesystemName": "*Name*",
The name of the file system to which the storage pool belongs.

"totalDataInKB": "*Size*",
The total data size in KB.

"freeDataInKB": "*size*",
The size of free space in KB.

"totalMetaInKB": "*size*",
The size of metadata in KB.

Examples

The following example gets information about the storage pool `System` in the file system `objfs`.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/objfs/pools/System'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  },
  "storagePool" : [ {
    "storagePoolName" : "system",
    "filesystemName" : "objfs",
    "totalDataInKB" : 0,
    "freeDataInKB" : 0,
    "totalMetaInKB" : 10485760,
    "freeMetaInKB" : 8889216
  } ]
}
```

Related reference

“mmlspool command” on page 528
Displays information about the known storage pools.

Filesystems/{filesystemName}/pools: GET

Gets the list of storage pools in a specific file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET filesystems/{filesystemName}/pools` request gets information about all the storage pools available in a particular file system. For more information about the fields in the data structures that are returned, see the topic "mmlspool command" on page 528.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/fileSystemName/
pools
```

where

filesystems
Specifies file system as the resource of the GET call. Required.

filesystemName
The particular file system in which you want to get information. Required.

pools
The list of storage pools in the file system. Required.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 126. List of parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	

Request data

No request data.

Response data

```
{
  "status" : {
    "code" : The HTTP status code,
    "message" : "The Return Status."
  },
  "storagePool" : [ {
    "filesystemName" : "The file system name",
    "storagePoolName" : "The storage pool name"
  }, {
    "filesystemName" : "The file system name",
    "storagePoolName" : "The file system name"
  } ]
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"code": "*ReturnCode*"
The return code.

"message": "*ReturnMessage*",
The return message.

"storagePool":
An array of elements that describe the storage pools within a file systems. Each element describes one file system and the storage pool within it.

"filesystemName":"*The File system name*",
The name of the file system.

"storagePoolName":"*The storage pool name*"
The name of the storage pool within the particular file system.

Examples

The following example gets information about storage pool name for a particular file system.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/objfs/pools'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  },
  "storagePool" : [ {
    "filesystemName" : "objfs",
    "storagePoolName" : "data"
  }, {
    "filesystemName" : "objfs",
    "storagePoolName" : "system"
  } ]
}
```

Related reference

“mmlspool command” on page 528
Displays information about the known storage pools.

Filesystems/{filesystemName}/symlink/{linkpath}: POST

Create a symlink for a path from the file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `filesystems/{filesystemName}/symlink/{linkPath}` request creates a symlink for a path from a specific file system.

Request URL

```
https://management API host:port/scalemgmt/v2/filesystems/filesystemName/symlink/linkPath
```

where:

filesystem/filesystemName
Specifies the name of the file system to which the symlink belongs. Required.

symlink/linkPath
Specifies the symlink path relative to the file system's mount point. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 127. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name.	Required.
linkpath	The symlink path relative to the fileset path.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation. This is the target path of the symlink relative to the file system's mount point or fileset path.	Required.

Request data

```
{
"filesystemName": "File system name",
"filesetName": "Fileset name",
"relativePath": "Relative path"
    }
```

"filesystemName": "*File system name*"
The name of the file system.

"filesetName": "*Fileset name*"
The name of the fileset.

"relativePath": "*Relative path*"
The target path of the symlink relative to the file system's mount point.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
   ],
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to create a symlink for the file system `gpfs0`.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
-d '{
  "filesystemName": "gpfs0",
   "filesetName": "fset1",
   "relativePath": "mydir"
}' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/symlink/mydir'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000005,
    "status" : "RUNNING",
    "submitted" : "2017-03-14 15:59:34,180",
    "completed" : "N/A",
    "request" : {
      "data" : {
                "filesystemName": "gpfs0",
                "filesetName": "fset1",
                "relativePath": "mydir"
                },
      "type" : "POST",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/symlink/mydir"
    },
    "result" : { }
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing"
  }
}
```

Filesystems/{filesystemName}/symlink/{path}: DELETE

Removes a symlink from a file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `DELETE filesystems/{filesystemName}/symlink/{path}` request removes a symlink from a file system.

Request URL

```
https://management API host:port/scalemgmt/v2/filesystems/filesystemName/symlink/path
```

where:

filesystems/filesystemName
Specifies the name of the file system from which the symlink must be removed. Required.

symlink/path
The symlink path relative to the file system's mount point.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 128. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name	Required.
path	The symlink path relative to the file system's mount point.	Required.

Request data

None.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
```

```
            "stdout":"String",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted":"Time",
        "completed":Time",
        "status":"Job status",
        }
 ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to remove symlink from the file system `gpfs0`.

Request data:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/symlink/myDir1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000006,
    "status" : "RUNNING",
    "submitted" : "2017-03-14 16:05:30,960",
    "completed" : "N/A",
    "request" : {
      "type" : "DELETE",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/symlink/myDir1"
    "result" : { }
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing"
  }
}
```

Filesystems/{filesystemName}/unmount: PUT

Performs unmount operation of a specific file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `filesystems/filesystemName/unmount` request unmounts a file system. For more information about the fields in the data structures that are returned, see "mmmount command" on page 545.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/FileSystemName/
unmount
```

where

filesystems/*filesystemName*
The file system that is going to be unmounted. Required.

unmount
Specifies the action to be performed on the file system. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 129. List of parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
          "nodes":"Node name} ",
         "remoteCluster": "Remote cluster name",
         "force": "True | False"
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"nodes":"*Node name*"
The node on which the file system must be unmounted.

"remoteCluster": "*Remote cluster name*"
Remote cluster name from where the file system must be unmounted.

"force": "*True | False*"
Forces the unmount to take place even though the file system may be still in use.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
   ],
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":"*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example how to unmount the file system `gpfs0`.

Request data:

```
{
  "nodes": "testnode-11",
  "remoteCluster": "myRemoteClusterName",
  "force": false
}
```

Corresponding request URL:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
-d -d '{ \
   "nodes": "testnode-11", \
   "remoteCluster": "myRemoteClusterName", \
   "force": false \  'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/unmount'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000002,
    "status" : "RUNNING",
    "submitted" : "2017-03-14 15:50:00,493",
    "completed" : "N/A",
    "request" : {
      "data" : {
        "entries" : [
      {
          "type" : "allow",
          "who" : "special:owner@",
          "permissions" : "rwmxDaAnNcCos",
          "flags" : ""
        },
         {
          "type" : "allow",
          "who" : "special:group@",
          "permissions" : "rxancs",
          "flags" : ""
```

```
        },
        {
          "type" : "allow",
          "who" : "special:everyone@",
          "permissions" : "rxancs",
          "flags" : ""
        },
        {
          "type" : "allow",
          "who" : "user:scalemgmt",
          "permissions" : "rxancs",
          "flags" : "fd"
        } ],
        "type" : "NFSv4"
      },
      "type" : "PUT",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/unmount"
    },
    "result" : { }
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing"
  }
}
```

“mmmount command” on page 545
Mounts GPFS file systems on one or more nodes in the cluster.

Filesystems/{filesystemName}/watch: PUT

Enables or disables clustered watch for a file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `filesystems/filesystemName/watch` request enables or disables clustered file system watch. For more information about the fields in the data structures that are returned, see “mmwatch command” on page 764.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/FileSystemName/
watch
```

where

filesystems/*filesystemName*
: The file system for which clustered watch needs to be enabled.

watch
: Specifies the action to be performed on the file system.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 130. List of parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{

  "action": "Enable or disable file system watch",
  "watchfolderConfig":
  {
    "description": "Description",
    "eventTypes": "Event types",
    "eventHandler": "Event handler",
```

```
    "sinkBrokers": "Sink broker",
    "sinkTopic": "Sink topic",
    "sinkAuthConfig": "Sink authentication config"
  },
  "watchId": "Watch ID"
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"action": "*Enable or disable file system watch*"
Whether to enable or disable file system watch. Possible values are `enable | disable`.

"watchfolderConfig":

"description": "*Description*"
Description of the clustered watch.

"eventTypes": "*Event types*"
Types of events that need to be watched.

"eventHandler": "*Event handler*"
Type of event handler. Only Kafkasink is supported as the evens handler.

"sinkBrokers": "*Sink broker*"
Includes a comma-separated list of broker:port pairs for the sink Kafka cluster, which is the external Kafka cluster where the events are sent.

"sinkTopic": "*Sink topic*"
The topic that producers write to in the sink Kafka cluster.

"sinkAuthConfig": "*Sink authentication config*"
The path to the sink authentication configuration file.

watchId": "*Watch ID*"
Remote cluster name from where the file system must be watched.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
   ],
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":"*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example how to enable the clustered file system watch for the file system `gpfs0`.

Request data:

```
{
  "action": "enable",
  "watchfolderConfig": {
    "description": "description",
    "eventTypes": "IN_ACCESS,IN_ATTRIB",
    "eventHandler": "kafkasink",
    "sinkBrokers": "Broker1:Port,Broker2:Port",
    "sinkTopic": "topic",
    "sinkAuthConfig": "/mnt/gpfs0/sink-auth-config"
  },
  "watchId": "string"
}
```

Corresponding request URL:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
-d '{ \
```

```
   "action": "enable", \
   "watchfolderConfig": { \
     "description": "description", \
     "eventTypes": "IN_ACCESS,IN_ATTRIB", \
     "eventHandler": "kafkasink", \
     "sinkBrokers": "Broker1:Port,Broker2:Port", \
     "sinkTopic": "topic", \
     "sinkAuthConfig": "/mnt/gpfs0/sink-auth-config" \
   }, \
   "watchId": "string" \
 }' 'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/watch'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": ["''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": ["''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": ["''EFSSG0740C There are not enough resources available to create
                   a new independent file set.'',  ...]",
        "stdout": ["''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "PUT",
        "url": "/scalemgmt/v2/filesystems/gpfs0/watch",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

“mmwatch command” on page 764
Administers clustered watch folder watches.

Filesystems/{filesystemName}/watches: GET

Gets a list of clustered watches in a file system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET filesystems/{filesystemName}/watches` request gets a list of clustered watches in a file system. For more information about the fields in the data structures that are returned, see “mmwatch command” on page 764.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/filesystems/
{filesystemName}/watches
```

where

filesystems/filesystemName
Specifies the file system to which the watches belong.

watches
Specifies clustered watches as the target of the GET request.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 131. List of parameters

Parameter name	Description and applicable keywords	Required/optional
filesystemName	The file system name. You can also use keywords such as :all:, :all_local:, or :all_remote:	Required.
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   },
 "paging":
  {
    "next": "URL",
    "fields": "Fields",
    "filter": "Filters",
    "baseUrl": "Base URL",
    "lastId": "ID of the last element",
  },
   watches: [
      {
      "oid": OID,
      "id": "ID of the file system watch",
      "device": "Device name",
      "fileset": "Fileset name",
      "path": "Path of the clustered watch",
      "type": "Type of the clustered watch",
      "events": "List of watched events",
      "eventHandler": "Event handler. Only Kafkasink is supported",
      "sinkBrokers": "List of sing brokers",
      "sinkTopic": "Sink topic",
      "degraded": "Degraded status of the clustred watch",
      "state": "The state of the clustered watch",
      "nodeStatus": [
        {
          "node": "Cluster node name",
          "status": "Clustered watch status on the node",
          "processType": "Process type"
        }
      ],
      "description": "description"
    }
   ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging":
Paging details.

"next": "*URL*",
The URL to retrieve the next page. Paging is enabled when more than 1000 objects would be returned by the query.

"fields": "*Fields*",
The fields used in the original request.

"filter": "*Filters*",
The filter used in the original request.

"baseUrl": "*Base URL*",
The URL of the request without any parameters.

"lastId": "*ID of the last element*",
The ID of the last element that can be used to retrieve the next elements.

"watches":
An array of elements that describe one watch.

"oid": *OID*
Optional ID.

"id": "*ID*"
ID of the file system watch.

"device": "*Device name*"
The device on which the clustered watch is enabled.

"fileset": "*Fileset name*"
The fileset on which the clustered watch is enabled.

"path": "*Path*"
Path of the clustered watch.

"type": "*Type of the clustered watch*"
Possible values are: `FILESYSTEM, INODESPACE, INODE, FILESET, INCOMPLETE, FSYS, FSET`

"events": "*Events*"
List of events that are watched.

"eventHandler": "*Event handler*"
Only Kafkasink is supported.

"sinkBrokers": "*List of sink brokers*"
Includes a comma-separated list of broker:port pairs for the sink Kafka cluster, which is the external Kafka cluster where the events are sent.

"sinkTopic": "*Sink topic*"
The topic that producers write to in the sink Kafka cluster.

"degraded": "*Degraded staus*"
Degraded status of the clustered watch.

"state": "*State*"
The state of the clustered watch.

description": "*Description*"
Description of the clustered watch.

"nodeStatus":"*Node status*"
Details of the status of the node.

"node": "*Node name*"
Name of the node.

"status": "*Clustered watch status on the node*"

"processType": "*Process type*"
Process type.

Examples

The following example gets information about the clustered watches in the file system `gpfs0`.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/filesystems/gpfs0/watches'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
```

```
    "code": 200,
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/filesystems/gpfs0/filesets?lastId=10001",
    "fields": "period,restrict,sensorName",
    "filter": "usedInodes>100,maxInodes>1024",
    "baseUrl": "/scalemgmt/v2/perfmon/sensor/config",
    "lastId": 10001
  },
  "watches": [
    {
      "oid": 0,
      "id": "CLW1595574252",
      "device": "gpfs0",
      "fileset": "fset1",
      "path": "/mnt/gpfs0",
      "type": "INODE",
      "events": "[\"IN_ACCESS\", \"IN_ATTRIB\"]",
      "eventHandler": "kafkasink",
      "sinkBrokers": "[\"198.51.100.5:8000\"]",
      "sinkTopic": "topic1",
      "degraded": false,
      "state": "Active",
      "nodeStatus": [
        {
          "node": "node1",
          "status": "status",
          "processType": "processType"
        }
      ],
      "description": "description"
    }
  ]
}
```


Related reference

“mmwatch command” on page 764
Administers clustered watch folder watches.

Gnr/clustermgmt/nodes/{names}/state: GET

Gets the status of the GPFS daemon on a specified node of the cluster.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET gnr/clustermgmt/nodes/{names}/state` request gets the details of the GPFS daemon on the node that is specified in the request parameter. For more information about the fields in the data structures that are returned, see the **mmgetstate** command in *IBM Spectrum Scale: Command and Programming Reference* in the IBM Spectrum Scale documentation.

Request URL

```
https://IP address of API server:<port>/scalemgmt/v2/gnr/clustermgmt/nodes/{names}/state
```

where

nodes
: Specifies the resource of this GET call. Required.

names
: Specifies the node names. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request data

No request data.

Response data

```
{
  "status": {
    "code": Status code,
    "message": "Status message"
  },
  "nodeState": [
    {
      "nodeName": "Node name",
      "number": Node number,
      "state": "UNKNOWN | ACTIVE | ARBITRATING | DOWN ",
      "quorum": Node quorum,
      "quorumUp": Number of quorum nodes up,
      "totalNode": Total number of nodes,
      "remarks": "Comments",
      "daemonNodeName": "Name of node"
      "daemonShortName": "Shortname of node"
      "cnfsState": "The GPFS daemon state"
    }
  ]
}
```

For more information about the fields in the following data structures, see the links at the end of the topic.

"status":
: Return status.

"message": "*ReturnMessage*"
The return message.

"code": *ReturnCode*
The return code.

"nodeState"
The state of the Nodes.

"nodeName":"*Node name*"
The name of the node.

"number": *Node number*
Specifies the number of the node.

"state":"*UNKNOWN | ACTIVE | ARBITRATING | DOWN* "
The status of GPFS daemon.

UNKNOWN
The state cannot be detected through the API request.

ACTIVE
The GPFS daemon is ready for operation.

ARBITRATING
A node is trying to form a quorum with the other available nodes.

DOWN
The GPFS daemon is not running on the node or is recovering from an internal error.

"quorum":"*quorum nodes*"
The number of quorum nodes.

"quorumUp":"*quorum nodes that are up* "
The number of quorum nodes that are up.

"totalnode":"*All nodes*"
The total number of nodes.

"remarks":"*comments*"
The additional comments on the node status.

"daemonNodeName":"*Node name*"
The name of the daemon node.

"daemonShortName":"*Node shortname*"
The shortname of the daemon node.

"cnfsState":"*GPFS daemon state*"
The state of the GPFS daemon.

Examples

The following example gets information on the status of the GPFS daemon on the node `hciece01`.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/gnr/clustermgmt/nodes/hciece01/state
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": 200,
    "message": "..."
```

```
  },
  "nodeStates": [
    {
      "nodeName": "hciece01",
      "daemonNodeName": "rhel-41.openstacklocal",
      "daemonShortName": "rhel-41",
      "number": 0,
      "state": "UNKNOWN",
      "quorum": 3,
      "quorumUp": 2,
      "totalNode": 3,
      "remarks": " ",
      "cnfsState": " "
    }
  ]
}
```

Related information:

mmgetstate command in the IBM Spectrum Scale documentation.

Info: GET

Gets information about IBM Spectrum Scale management API version and supported commands.

Availability

Available on all IBM Spectrum Scale editions.

Description

The GET `info` request gets information about IBM Spectrum Scale management API. The information includes the version number, the resources that are supported, and the HTTP methods that are supported for each resource.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/info
```

where

`info`
Is the information resource.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request data

No request data.

Response data

```
{
     "status": {
        "message":"ReturnMessage",
        "code":"ReturnCode"
       }
    "info":
         {
        "restAPIVersion":"Version",
        "name": "API name",
        "path": "Resource path"
         }
}
```

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"info":
Information about the REST API server.

"restAPIVersion":"*Version*"
The IBM Spectrum Scale management API version.

"name":"*APIName*"
The name of the API, such as "Spectrum Scale REST Management".

"path": "*Resource path*
Supported resources and elements.

Examples

The following example gets information about the REST API server.

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/info'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
"info" : {
"name" : "Spectrum Scale REST Management",
"restApiVersion" : "2.1.0",
"serverVersion" : "5.0.0-0_170818.165000",
"paths" : {
"/access" : [ "GET", "POST" ],
"/access/status" : [ "GET" ],
"/ces/addresses" : [ "GET" ],
"/ces/addresses/{cesAddress}" : [ "GET" ],
"/ces/services" : [ "GET" ],
"/ces/services/{service}" : [ "GET" ],
"/cluster" : [ "GET" ],
"/config" : [ "GET" ],
"/filesystems" : [ "GET" ],
"/filesystems/{filesystemName}" : [ "GET" ],
"/filesystems/{filesystemName}/acl/{path}" : [ "GET", "PUT" ],
"/filesystems/{filesystemName}/afm/state" : [ "GET" ],
"/filesystems/{filesystemName}/disks" : [ "GET" ],
"/filesystems/{filesystemName}/disks/{diskName}" : [ "GET" ],
"/filesystems/{filesystemName}/filesets" : [ "GET", "POST" ],
"/filesystems/{filesystemName}/filesets/{filesetName}" : [ "GET", "PUT", "DELETE" ],
"/filesystems/{filesystemName}/filesets/{filesetName}/link" : [ "POST", "DELETE" ],
"/filesystems/{filesystemName}/filesets/{filesetName}/psnaps" : [ "POST" ],
"/filesystems/{filesystemName}/filesets/{filesetName}/psnaps/{snapshotName}" : [ "DELETE" ],
"/filesystems/{filesystemName}/filesets/{filesetName}/quotas" : [ "GET", "POST" ],
"/filesystems/{filesystemName}/filesets/{filesetName}/snapshots" : [ "GET", "POST" ],
"/filesystems/{filesystemName}/filesets/{filesetName}/snapshots/{snapshotName}" : [ "GET",
"DELETE" ],
"/filesystems/{filesystemName}/owner/{path}" : [ "GET", "PUT" ],
"/filesystems/{filesystemName}/quotas" : [ "GET", "POST" ],
"/filesystems/{filesystemName}/snapshots" : [ "GET", "POST" ],
"/filesystems/{filesystemName}/snapshots/{snapshotName}" : [ "GET", "DELETE" ],
"/info" : [ "GET" ],
"/jobs" : [ "GET" ],
"/jobs/{jobId}" : [ "GET", "DELETE" ],
"/nfs/exports" : [ "GET", "POST" ],
"/nfs/exports/{exportPath}" : [ "GET", "DELETE", "PUT" ],
"/nodeclasses" : [ "GET", "POST" ],
"/nodeclasses/{nodeclassName}" : [ "GET", "DELETE", "PUT" ],
"/nodes" : [ "GET", "POST" ],
"/nodes/{name}" : [ "GET", "DELETE" ],
"/nodes/{name}/health/events" : [ "GET" ],
"/nodes/{name}/health/states" : [ "GET" ],
"/nsds" : [ "GET" ],
"/nsds/{nsdName}" : [ "GET" ],
"/perfmon/data" : [ "GET" ],
"/smb/shares" : [ "GET", "POST" ],
"/smb/shares/{shareName}" : [ "GET", "DELETE", "PUT" ],
"/thresholds" : [ "GET", "POST" ],
"/thresholds/{name}" : [ "GET", "DELETE" ]
}
},
"status" : {
"code" : 200,
```

```
"message" : "The request finished successfully"
}
}
```

Jobs: GET

Gets details about the asynchronous jobs.

Availability

Available on all IBM Spectrum Scale editions.

Description

The GET `jobs` request gets information about the asynchronous jobs.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/jobs
```

where

jobs
Specifies running jobs as the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 132. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "paging":
    {
    "next": "URL"
    },
   jobs: [
      {
```

```
        "result":"",
          {
            "commands":"String",
            "progress":"String,
            "exitCode":"Exit code",
            "stderr":"Error",
            "stdout":"String",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted":"Time",
        "completed":Time",
        "status":"Job status",
        }
       ],
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example gets information about the jobs.

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/jobs
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000001,
    "status" : "COMPLETED",
    "submitted" : "2017-03-21 15:49:20,100",
    "completed" : "2017-03-21 15:49:24,251"
  }, {
    "jobId" : 1000000000002,
    "status" : "COMPLETED",
    "submitted" : "2017-03-21 15:52:13,776",
    "completed" : "2017-03-21 15:52:16,030"
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully"
  }
}
```

Using the field parameter ":all:" returns entire details of the jobs. For example:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/jobs?fields=:all:'
```

```
{
  "jobs" : [ {
    "jobId" : 1000000000001,
    "status" : "COMPLETED",
    "submitted" : "2017-03-21 15:49:20,100",
    "completed" : "2017-03-21 15:49:24,251",
    "request" : {
      "data" : {
        "snapshotName" : "mySnap1"
      },
      "type" : "POST",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/snapshots"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmcrsnapshot 'gpfs0' 'mySnap1' " ],
      "stdout" : [ "EFSSG0019I The snapshot mySnap1 has been successfully created." ],
      "stderr" : [ ],
      "exitCode" : 0
    }
  }, {
    "jobId" : 1000000000002,
    "status" : "COMPLETED",
    "submitted" : "2017-03-21 15:52:13,776",
    "completed" : "2017-03-21 15:52:16,030",
    "request" : {
      "data" : {
        "snapshotName" : "mySnap1"
```

```
      },
      "type" : "POST",
      "url" : "/scalemgmt/v2/filesystems/gpfs0/filesets/fset1/snapshots"
    },
    "result" : {
      "progress" : [ ],
      "commands" : [ "mmcrsnapshot 'gpfs0' 'mySnap1' -j 'fset1' " ],
      "stdout" : [ "EFSSG0019I The snapshot mySnap1 has been successfully created." ],
      "stderr" : [ ],
      "exitCode" : 0
    }
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully"
  }
}
```

Jobs/{jobId}: DELETE

Cancels an asynchronous job.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `DELETE Jobs/{jobId}` request cancels a job.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/jobs/{jobId}
```

where

jobs/{jobId}
Specifies the job to be canceled. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 133. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
jobId	Unique identifier of the asynchronous job.	Required.
force	Forces cancellation of jobs. Possible values are **true** or **false**. If the value selected is **true**, it forces job cancellation	Optional.

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
```

```
            "stdout":"String",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted":"Time",
        "completed":Time",
        "status":"Job status",
        }
      ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED | CANCELLING | CANCELLED"
Status of the job.

Examples

The following API command cancels the job *1234*.

Request URL:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/jobs/1234'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": "[''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": "[''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": "[''EFSSG0740C There are not enough resources available to create
                  a new independent file set.'',  ...]",
        "stdout": "[''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "DELETE",
        "url": "/scalemgmt/v2/jobs/1234",
        },
      "jobId": "1234",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

Jobs/{jobID}: GET

Gets details about an asynchronous job.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET jobs/jobID` request gets information about the asynchronous job that is specified in the request.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/jobs/jobID
```

where

jobs
Specifies running jobs as the resource of this GET call. Required.

jobs/jobID
Specifies the job about which you need to get information. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 134. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
jobId	Unique identifier of the asynchronous job.	Required.
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "paging":
    {
    "next": "URL"
    },
```

```
jobs: [
   {
      "result":"",
        {
          "commands":"String",
          "progress":"String,
          "exitCode":"Exit code",
          "stderr":"Error",
          "stdout":"String",
        },
      "request":" ",
        {
          "type":"{GET | POST | PUT | DELETE}",
          "url":"URL",
          "data":"",
        }
      "jobId":"ID",
      "submitted":"Time",
      "completed":Time",
      "status":"Job status",
      }
     ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Known limitation

If there are more than one GUI nodes in a cluster, you need to use the filter parameter to get the details of the particular job. For example, if a job is submitted on GUI2 and it returns a jobId=2000000000003. If you query GUI1 for that job ID using */jobs/2000000000003*, it returns "Invalid request" because it cannot find the job since it was submitted to the other GUI. A workaround is to use "*/jobs?filter=jobId=2000000000003*" instead of "*/jobs/2000000000003*" in the request URL to retrieve the job correctly from the other GUI.

Examples

The following example gets information about the job 12345.

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/jobs/12345'
```

Note: If there are more than one GUI nodes in the cluster, use the following request URL to retrieve the details of the job:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/jobs?filter=jobId=12345'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/filesystems/gpfs0/filesets?lastId=1001"
  },
  "jobs": [
    {
      "result": {
        "commands": "[''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": "[''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": "[''EFSSG0740C There are not enough resources available to create
                  a new independent file set.'',  ...]",
        "stdout": "[''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "GET",
        "url": "/scalemgmt/v2/filesystems/gpfs0/filesets",
        "data": "{\"config\":{\"filesetName\":\"restfs1001\",\"owner\":\"root\",\"path\":
                 \"/mnt/gpfs0/rest1001\",\"permissions\":\"555\"}"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

NFS/exports: GET

Gets information about NFS exports configured in the system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The GET `nfs/exports` request gets information about NFS exports. For more information about the fields in the data structures that are returned, see “mmnfs command” on page 560.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nfs/exports
```

where

`nfs/exports`
Specifies NFS export as the resource. Required.

Request headers

```
Accept: application/json
```

Request data

No request data.

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 135. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Response data

The following list of attributes are available in the response data:

```
{
   "status":
     {
      "code": ReturnCode
      "message": "ReturnMessage",
     }
  "NfsExports":
    [
```

```
      {
         "filesystemName": "FileSystemName"
         "path": "Path",
         "delegations": "Delegations",
         "pseudoPath": "Pseudo path",
      }],
   nfsClients":[
      {
         "clientName": "String"
         "exportID": "Path",
         "access_type": "none | RW | RO | MDONLY | MDONLY_RO",
         "squash": "root | root_squash | all | all_squash | allsquash | no_root_squash | none"
         "anonUid": "ID",
         "anonGid": "ID",
         "privilegedPort": "True | False"
         "sectype": "sys | krb5 | krb5i | krb5p | None",
         "protocols": "NFS3 | NFS4",
         "transportProtocol": "TCP | UDP"
         "delegations": "none | read | write | readwrite",
         "manageGids": "True | False"
         "nfsCommit": "True | False",
      }
}
```

"status":
Return status.

"code": *ReturnCode*,
The HTTP status code that was returned by the request.

"message": "*ReturnMessage*"
The return message.

"exports":
An array of information about NFS exports. The array contains elements that describe the NFS exports. For more information about the fields in this structure, see the links at the end of this topic.

"filesystemName": "*FileSystemName*"
The name of the file system to which the export belongs.

"path": "*Path*",
Specifies the path for the export.

"pseudoPath": "*Pseudo path*"
Specifies the path name that is used by the NFSv4 client to locate the directory in the server's file system tree.

nfsClients

"clientName": "*String*"
The host name or IP address of the NFS server to be accessed.

"exportID": "*Path*"
The internal unique ID of the export.

"access_type": "*none | RW | RO | MDONLY | MDONLY_RO*"
Specifies the type of the access for the client.

"squash": "*root | root_squash | all | all_squash | allsquash | no_root_squash | none*"
Specifies whether the squashing mechanism is applied to the connecting client.

"anonUid": "*ID*"
This option explicitly sets the UID of the anonymous account.

"anonGid": "*ID*"
This option explicitly sets the GID of the anonymous account.

"privilegedPort": "*True | False*"
This option to specify whether a client that uses an ephemeral port must be rejected.

"sectype": "*sys | krb5 | krb5i | krb5p | None*"
The supported authentication method for the client.

"protocols": "*NFS3 | NFS4* "

Supported NFS protocol version.

"transportProtocol": "*TCP | UDP*"
Specifies the transport layer.

"delegations": "*none | read | write | readwrite*"
Specifies what delegate file operations are permitted.

"manageGids": "*True | False*"
Whether this NFS client is allowed to manage GIDs.

"nfsCommit": "*True | False*"

Whether to commit the transmitted data on the server side.

Examples

The following example gets information about the NFS exports that are configured in the system:

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/nfs/exports'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "nfsexports" : [ {
        "path" : "/mnt/gpfs0"
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully"
  }
}
```

Using the field parameter ":all:" returns entire details of the NFS exports as shown in the following example:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/nfs/exports?fields=:all:'
```

```
{
  "nfsexports" : [ {
       "delegations" : "NONE",
      "filesetName" : "",
      "filesystemName" : "gpfs0",
      "path" : "/mnt/gpfs0"
      "pseudoPath" : "/mnt/test"
    },
    "nfsClients" : [ {
      "access_type" : "RW",
      "anonGid" : "-2",
      "anonUid" : "-2",
      "clientName" : "198.51.100.123",
      "delegations" : "NONE",
      "exportid" : "1",
      "manageGids" : false,
      "nfsCommit" : false,
      "privilegedPort" : false,
      "protocols" : "3,4",
      "sectype" : "SYS",
      "squash" : "ROOT_SQUASH",
      "transportProtocol" : "TCP"
    } ]
```

```
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully"
  }
}
```

Related reference

“mmnfs command” on page 560
Manages NFS exports and configuration.

NFS/exports: POST

Creates an NFS export.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `nfs/export` request creates a new NFS export. For more information about the fields in the data structures that are returned, see “mmnfs command” on page 560.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nfs/exports
```

where

`nfs/exports`
Specifies the NFS export as the resource. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 136. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
        "path": "Path",
        "pseudoPath": "Pseudo path",
        "nfsclients": "Client details"
}
```

"path": "*Path* "
Specifies the path of the NFS export.

"pseudoPath": "*Pseudo path*"
Specifies the path name that is used by the NFSv4 client to locate the directory in the server's file system tree.

nfsClients
Details of the NFS clients.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example creates an NFS export.

Request URL:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
-d '{
  "path": "/mnt/gpfs0/fset1", "pseudoPath": "/mnt/test/",
  "nfsClients": [ "198.51.100.111:443(access_type=ro,squash=no_root_squash)",
     "198.51.100.110:443(access_type=rw,squash=no_root_squash)" ]
}'
'https://198.51.100.1:443/scalemgmt/v2/nfs/exports'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": ["''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": ["''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": ["''EFSSG0740C There are not enough resources available to create
                  a new independent file set.'',  ...]",
        "stdout": ["''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "POST",
        "url": "/scalemgmt/v2/filesystems/gpfs0/filesets",
        "data": "{
          "path": "/mnt/gpfs0/fset1",
           "pseudoPath": "/mnt/test/",
           "nfsClients": [ ''198.51.100.111:443(access_type=ro,squash=no_root_squash)'',
                           ''198.51.100.110:443(access_type=rw,squash=no_root_squash)'' ]}"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

Related reference

“mmnfs command” on page 560
Manages NFS exports and configuration.

NFS/exports/{exportPath}: GET

Gets information about an NFS export that is configured in the system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET nfs/exports/exportPath` request gets information about an NFS export. For more information about the fields in the data structures that are returned, see “mmnfs command” on page 560.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nfs/exports
```

where

nfs/exports
Specifies NFS export as the resource. Required.

exportPath
Specifies the NFS export about which you need the details. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 137. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
exportPath	The path of the NFS export. The export path is relative to file system's mount point. The path of the file or directory is specified with forward slashes (/). For example: mnt/gpfs0/rest01. The forward slashes in the path are encoded to %2F in the request URL.	Required.

Request data

No request data.

Response data

The following list of attributes are available in the response data:

```
{
   "status":
     {
      "code": ReturnCode
      "message": "ReturnMessage",
     }
  "exports":
    {
     "config":[
      {
         "filesystemName": "FileSystemName"
         "path": "Path",
         "delegations": "Delegations",
         "pseudoPath": "Pseudo path",
      }],
   nfsClients":[
      {
         "clientName": "String"
         "exportID": "Path",
         "access_type": "none | RW | RO | MDONLY | MDONLY_RO",
         "squash": "root | root_squash | all | all_squash | allsquash | no_root_squash | none"
         "anonUid": "ID",
         "anonGid": "ID",
         "privilegedPort": "True | False"
         "sectype": "sys | krb5 | krb5i | krb5p | None",
         "protocols": "NFS3 | NFS4",
         "transportProtocol": "TCP | UDP"
         "delegations": "none | read | write | readwrite",
         "manageGids": "True | False"
         "nfsCommit": "True | False",
      }
}
```

"status":
Return status.

"code": *ReturnCode*,
The HTTP status code that was returned by the request.

"message": "*ReturnMessage*"
The return message.

"exports":
An array of information about NFS exports. The array contains elements that describe the NFS exports. For more information about the fields in this structure, see the links at the end of this topic.

config":
Manages NFS configuration in a cluster.

"filesystemName": "*FileSystemName*"
The name of the file system to which the export belongs.

"path": "*Path*",
Specifies the path for the export.

"delegations": "*Delegations*"
Specifies what delegate file operations are permitted.

"pseudoPath": "*Pseudo path*"
Specifies the path name that is used by the NFSv4 client to locate the directory in the server's file system tree.

nfsClients

"clientName": "*String*"
The host name or IP address of the NFS server to be accessed.

"exportID": "*Path*"
The internal unique ID of the export.

"access_type": "*none | RW | RO | MDONLY | MDONLY_RO*"
Specifies the type of the access for the client.

"squash": "*root | root_squash | all | all_squash | allsquash | no_root_squash | none*"
Specifies whether the squashing mechanism is applied to the connecting client.

"anonUid": "*ID*"
This option explicitly sets the UID of the anonymous account.

"anonGid": "*ID*"
This option explicitly sets the GID of the anonymous account.

"privilegedPort": "*True | False*"
This option to specify whether a client that uses an ephemeral port must be rejected.

"sectype": "*sys | krb5 | krb5i | krb5p | None*"
The supported authentication method for the client.

"protocols": "*NFS3 | NFS4* "

Supported NFS protocol version.

"transportProtocol": "*TCP | UDP*"
Specifies the transport layer.

"delegations": "*none | read | write | readwrite*"
Specifies what delegate file operations are permitted.

"manageGids": "*True | False*"
Whether this NFS client is allowed to manage GIDs.

"nfsCommit": "*True | False*"

Whether to commit the transmitted data on the server side.

Examples

The following example gets information about the NFS export `/mnt/gpfs0/fset1` that is configured in the system.

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/nfs/exports/%2Fmnt%2Fgpfs0%2Ffset1
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/nfs/exports/mnt%2Fgpfs0%2Ffset1?lastId=1001"
  },
  "exports": [
    {
      "config": {
        "filesystemName": "gpfs0",
        "path": "/mnt/gpfs0/fset1",
        "delegations": "NONE",
        "pseudoPath" : "/mnt/test"
      },
      "nfsClients": [
        {
          "clientName": "198.51.100.8",
          "exportid": "1",
          "access_type": "RO",
          "squash": "root_squash",
          "anonUid": "1",
```

```
          "anonGid": "1",
          "privilegedPort": "false",
          "sectype": "sys",
          "protocols": "NFS4",
          "transportProtocol": "TCP",
          "delegations": "none",
          "manageGids": "yes",
          "nfsCommit": "This is a comment"
        }
      ]
    }
  ]
}
```

Related reference

“mmnfs command” on page 560
Manages NFS exports and configuration.

NFS/exports/{exportPath}: PUT

Modifies an existing NFS export.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `nfs/export/exportPath` request modifies an existing NFS export. For more information about the fields in the data structures that are returned, see “mmnfs command” on page 560.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nfs/exports/exportPath
```

where

nfs/exports
Specifies the NFS export as the resource. Required.

exportPath
Specifies the NFS export, which you need to modify. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 138. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
exportPath	The path of the NFS export. The export path is relative to file system's mount point. The path of the file or directory is specified with forward slashes (/). For example: mnt/gpfs0/rest01. The forward slashes in the path are encoded to %2F in the request URL.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
        "nfsadd": "New client",
```

```
          "nfschange": "Change export",
          "nfsremove": "Remove NFS client",
          "nfspostion": "NFS position"
}
```

Detailed explanation of the fields are given in the following list:

"nfsadd": "*New client*"
: Adds a new client declaration for the specified path. Client options can be a list of one or more client definitions.

"nfschange": "*Change export*"
: Modifies an existing client declaration for the specified path. Client options can be a list of one or more client definitions.

"nfsremove": "*Remove NFS client*"
: Removes the NFS client specified by client from the export configuration for the path.

"nfspostion": "*NFS position*"
: It reorders the client declaration section within the export declaration file. This option can be used only together with **--nfsadd** or **--nfschange**.

Response data

```
{
     "status": {
       "code":ReturnCode",
       "message":"ReturnMessage"
    },
    jobs: [
       {
          "result":"",
            {
              "commands":"String",
              "progress":"String,
              "exitCode":"Exit code",
              "stderr":"Error",
              "stdout":"String",
            },
          "request":" ",
            {
              "type":"{GET | POST | PUT | DELETE}",
              "url":"URL",
              "data":"",
            }
          "jobId":"ID",
          "submitted":"Time",
          "completed":Time",
          "status":"Job status",
          }
        ],
    }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
: Return status.

  "message": "*ReturnMessage*",
  : The return message.

  "code": *ReturnCode*
  : The return code.

"jobs":
: An array of elements that describe jobs. Each element describes one job.

  "result"

    "commands":"*String*"
    : Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example modifies the NFS export `/mnt/gpfs0/fset1` that is configured in the system.

Request URL:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
-d '{
  "nfsadd": "198.51.100.10:443(sectype=sys,krb5)",
  "nfschange": "198.51.100.11:443(sectype=sys,krb5)",
  "nfsremove": "198.51.100.21:443",
  "nfsposition": "1"
}' "https://198.51.100.1:443/scalemgmt/v2/nfs/exports/mnt%2Fgpfs0%2Ffs1"
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {},
      "request": {
        "type": "PUT",
        "url": "/scalemgmt/v2/nfs/exports/mnt%2Fgpfs0%2Ffs1",
        "data": "{
          "nfsadd": "198.51.100.10:443(sectype=sys,krb5)",
          "nfschange": "198.51.100.11:443(sectype=sys,krb5)",
          "nfsremove": "198.51.100.21:443",
          "nfsposition": "1""}"
      },
```

```
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

Related reference

“mmnfs command” on page 560
Manages NFS exports and configuration.

NFS/exports/{exportPath}: DELETE

Deletes an NFS export.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `DELETE nfs/exports/exportPath` command deletes the specified NFS export. For more information, see “mmnfs command” on page 560.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nfs/exports/exportPath
```

where:

nfs/exports
Specifies NFS export as the resource. Required.

exportPath
Specifies the NFS export to be deleted. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 139. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
path	The file path relative to file system's mount point. The path of the file or directory is specified with forward slashes (/). For example: `mnt/gpfs0/rest01`. The forward slashes in the path are encoded to "%2F" in the request URL.	Required.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
```

```
            "stderr":"Error",
            "stdout":"String",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted":"Time",
        "completed":Time",
        "status":"Job status",
        }
       ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example deletes the NFS export `/mnt/gpfs0/fset1`.

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/nfs/exports/%2Fmnt%2Fgpfs0%2Ffset1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {},
      "request": {
        "type": "DELETE",
        "url": "/scalemgmt/v2/nfs/exports/%2Fmnt%2Fgpfs0%2Ffset1",
        "data": "{}"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

Related reference

“mmnfs command” on page 560
Manages NFS exports and configuration.

Nodes/network: GET

Gets information on all networks that are configured and are operational for a group of nodes.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET nodes/network` request gets information on the networks that are configured and are operational on a group of nodes. For more information about the fields in the data structures that are returned, see “mmnetverify command” on page 548.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nodes/network
```

where

network
: Specifies the resource for which you need to get the details. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 140. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Optional.

Request data

The following list of attributes are available in the request data.

```
{
  "operation": "port|all|data|local|bandwidth",
  "node": "Node name",
  "targetNodes": "Target node names",
  "clusterName": "Cluster IP address",
  "configFile": "File path",
  "pingPacket": Size,
  "logFile": "File path",
  "addr": "subnet list",
  "verbose": true | false,
  "cesOverride": "CES-enabled ",
  "minBandwidth": Bandwidth size,
  "maxThreads": Number of threads
}
```

The details of the attributes are given in the following list.

"operation": "port|all|data|local|bandwidth"
The operations that are run against the target nodes to check network configuration and operation. You can specify more than one operation. Each operation is separated by a blank.

"node": "Node name"
The list of nodes on which to run the command. If you specify more than one node, the command is run on all the specified nodes in parallel. Each node tests all the specified target nodes. If you do not specify any value for this parameter, the command runs the operations only from the node where you enter the command.

"targetNodes": "Targetnode name"
The list of nodes that are the targets of the testing. If you do not specify this parameter, the command runs the operations against all the nodes in the cluster.

"targetNodes": "Targetnode name"
The list of nodes that are the targets for the testing. If you do not specify this parameter, the command runs the operations against all the nodes in the cluster.

"clusterName": "Cluster IP address"
The name of a remote cluster followed by at least one contact node identifier.

"configFile": "file path"
The path of a configuration file.

"pingPacket": "Size"
The size, in bytes, of the ICMP echo request packets that are sent between the local node and the target node during the `ping` test. The size must not be greater than the MTU of the network interface. If the MTU size of the network interface changes, for example to support jumbo frames, you can specify this parameter to verify that all the nodes in the cluster can handle the new MTU size.

"logFile": "File path"
The path of a file where the output messages from the network checks are stored. If you do not specify this parameter, messages are displayed only on the console.

"addr": "Subnet list"
The list of subnets that the command searches in a sequential order for a subnet that both the local node and the target node are connected to.

"verbose": true | false
Specifies whether verbose output messages are generated.

"cesOverride": "CES-enabled"
Specifies whether the nodes in the configuration are CES-enabled. This parameter overrides the requirement of the `protocol-ctdb` and the `protocol-object` network checks that the local node and the target nodes must be CES-enabled with the **mmchnode** command. This parameter is automatically set if a value is specified for the **configFile** parameter.

"minBandwidth": size
The minimum acceptable bandwidth for the data bandwidth check.

"maxThreads": size
The number of nodes that can run the command in parallel. The valid range is 1 - 64 and the default value is 32. For more information, see the "mmnetverify command" on page 548.

Response data

The following list of attributes are available in the response data:

```
{
"status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "mmnetverify": [
    {
      "localNode": "Node name",
      "targetNode": "Target node name",
      "test": "Test nodes",
```

```
      "data": "Data size",
      "status": "Network status"
    }
  ]
}
```

The details of the parameters are given in the following list:

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"mmnetverify"
The command that is run.

"localNode": "*Node name*"
The node from which a network test is run.

"targetNode": "*Target node name*"
The node against which a test is run.

"test": "*Test nodes*"
The nodes that are involved in the test, including both local nodes and target nodes.

"data": "*Data size*"
The size of data messages specified for network checks.

"status": "*Network status*"
The network status.

Examples

The following example gets information on the network configuration and operation .

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json' -d '{ \
   "operation": "all", \
   "node": "node1-11", \
   "targetNodes": "node1-12", \
   "clusterName": "9.114.204.107", \
   "configFile": "string", \
   "pingPacket": 0, \
   "logFile": "string", \
   "addr": "string", \
   "verbose": false, \
   "cesOverride": "string", \
   "minBandwidth": 0, \
   "maxThreads": 0 \
 }' 'https://198.51.100.1:443/scalemgmt/v2/nodes/network'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  {
    "mmnetverify": [
        {
            "data": "0",
            "localNode": "node1-11",
            "status": "0",
            "targetNode": "",
```

```
            "test": "interface"
        },
        {
            "data": "node1-11",
            "localNode": "node1-11",
            "status": "0",
            "targetNode": "node1-11",
            "test": "resolution"
        },
        {
            "data": "node1-11",
            "localNode": "node1-11",
            "status": "0",
            "targetNode": "node1-11",
            "test": "ping"
        },
        {
            "data": "node1-11",
            "localNode": "node1-11",
            "status": "0",
            "targetNode": "node1-11",
            "test": "shell"
        },
        {
            "data": "node1-11",
            "localNode": "node1-11",
            "status": "0",
            "targetNode": "node1-11",
            "test": "copy"
        }
        ]
        }
}
```

Related reference

“mmnetverify command” on page 548
Verifies network configuration and operation in a cluster.

Nodeclasses: GET

Gets details about node classes in the IBM Spectrum Scale cluster.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET nodeclasses` request gets information about all node classes and the nodes that are part of each node class. For more information about the fields in the data structures that are returned, see the topics “mmcrnodeclass command” on page 333, “mmdelnodeclass command” on page 379, “mmchnodeclass command” on page 251, and “mmlsnodeclass command” on page 520.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nodeclasses
```

where

nodeclasses
Specifies nodes classes as the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request data

No request data.

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 141. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Response data

```
{
  "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "paging":
    {
```

```
    "next": "URL"
    },
   nodeClasses: [
      {
          "nodeclassName": "Name",
          "members": "[node1Name, node2Name, node3Name]",
          "type": "SYSTEM | USER"
      }
        ]
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"nodeClasses":

"nodeclassName":"*Name*"
The name of the node class.

"members":"*List of nodes*"
The list of nodes that are part of the node class.

"type": "*SYSTEM | USER*"
Indicates whether the node class is system-defined or user-defined.

Examples

The following example gets information about the node classes that are available in the cluster.

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/nodeclasses'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

The *nodeclasses* array returns 10 objects in the following example. Each object contains details about one node class.

```
{
  "nodeclasses" : [ {
    "nodeclassName" : "aixNodes",
    "type" : "SYSTEM"
  }, {
    "nodeclassName" : "all",
    "type" : "SYSTEM"
  }, {
    "nodeclassName" : "cesNodes",
    "type" : "SYSTEM"
  }, {
    "nodeclassName" : "clientLicense",
    "type" : "SYSTEM"
  }, {
    "nodeclassName" : "clientNodes",
```

```
    "type" : "SYSTEM"
  }, {
    "nodeclassName" : "cnfsNodes",
    "type" : "SYSTEM"
  }, {
    "nodeclassName" : "disabledCnfsNodes",
    "type" : "SYSTEM"
  }, {
    "nodeclassName" : "enabledCnfsNodes",
    "type" : "SYSTEM"
  }, {
    "nodeclassName" : "GUI_MGMT_SERVERS",
    "type" : "USER"
  }, {
    "nodeclassName" : "GUI_SERVERS",
    "type" : "USER"
  }, ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully"
  }
}
```

“mmchnode command” on page 244
Changes node attributes.

“mmgetstate command” on page 431
Displays the state of the GPFS daemon on one or more nodes.

Nodeclasses: POST

Creates a user-defined node class in the IBM Spectrum Scale cluster.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `nodeclasses` request creates a node class and assign nodes to it. For more information about the fields in the data structures that are returned, see the topics “mmcrnodeclass command” on page 333, “mmdelnodeclass command” on page 379, “mmchnodeclass command” on page 251, and “mmlsnodeclass command” on page 520.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nodeclasses
```

where

nodeclasses
Specifies node class as the target. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 142. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
   nodeClasses:
      {
          "nodeclassName": "Name",
          "members": "[node1Name, node2Name, node3Name]",
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"nodeclassName":"*Name*"
The name of the node class.

"members":"*List of nodes*"
The list of nodes that are part of the node class.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example creates a node class named *cesNodes*.

Request URL:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
-d '{"nodeclassName": "cesNodes", "members": ['mari-11', 'cloudOne1']}'
'https://198.51.100.1:443/scalemgmt/v2/nodeclasses'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": "[''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": "[''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": "[''EFSSG0740C There are not enough resources available to create
                   a new independent file set.'',  ...]",
        "stdout": "[''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "POST",
        "url": "/scalemgmt/v2/nodeclasses",
        "data": "{
         "nodeclassName": "cesNodes",
          "members": "['mari-11', 'cloudOne1]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
```

```
    }
  ]
}
```

“mmchnode command” on page 244
Changes node attributes.

“mmgetstate command” on page 431
Displays the state of the GPFS daemon on one or more nodes.

Nodeclasses/{nodeclassName}: GET

Gets details about a specific node class in the IBM Spectrum Scale cluster.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET nodeclasses/{nodeclasseName}` request gets information about a specific and the nodes that are part of that node class. For more information about the fields in the data structures that are returned, see the topics “mmcrnodeclass command” on page 333, “mmdelnodeclass command” on page 379, “mmchnodeclass command” on page 251, and “mmlsnodeclass command” on page 520.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nodeclasses/nodeclassName
```

where

nodeclasses/nodeclassName
Specifies a particular nodes class as the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request data

No request data.

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 143. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.
nodeclassName	The name of the node class.	Required.

Response data

```
{
  "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
```

```
   "paging":
    {
    "next": "URL"
    },
   nodeClasses: [
      {
          "nodeclassName": "Name",
          "members": "[node1Name, node2Name, node3Name]",
          "type": "SYSTEM | USER"
      }
        ]
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"nodeClasses":

"nodeclassName":"*Name*"
The name of the node class.

"members":"*List of nodes*"
The list of nodes that are part of the node class.

"type": "*Type*
Indicates whether the node class is system-defined or user-defined.

Examples

The following example gets information about the node class *cesNodes*.

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/nodeclasses/cesNodes'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "nodeClasses": [
    {
      "nodeclassName": "cesNodes",
      "members": "[mari-12.localnet.com,mari-13.localnet.com]",
      "type": "SYSTEM"
    }
  ]
}
```

“mmchnode command” on page 244

Changes node attributes.

“mmgetstate command” on page 431
Displays the state of the GPFS daemon on one or more nodes.

Nodeclasses/{nodeclassName}: DELETE

Deletes a specific user-defined node class.

Availability

Available on all IBM Spectrum Scale editions.

Description

The DELETE nodeclass request deletes a specific node class. For more information about the fields in the data structures that are returned, see the topics “mmcrnodeclass command” on page 333, “mmdelnodeclass command” on page 379, “mmchnodeclass command” on page 251, and “mmlsnodeclass command” on page 520.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nodeclass/nodeclassName
```

where

nodeclasses/nodeclassName
Specifies the node class to be deleted. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 144. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
nodeclassName	The name of the node class to be deleted. You can also specify keywords such as :all:, :all_local:, or :all_remote:	Required.

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
```

```
            "stderr":"Error",
            "stdout":"String",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted":"Time",
        "completed":Time",
        "status":"Job status",
        }
      ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example deletes the a node class named *cesNodes*.

Request URL:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/nodeclasses/cesNodes'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": "[''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": "[''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": "[''EFSSG0740C There are not enough resources available to create
                  a new independent file set.'',  ...]",
        "stdout": "[''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "DELETE",
        "url": "/scalemgmt/v2/nodeclasses/cesNodes",
       },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

“mmchnode command” on page 244
Changes node attributes.

“mmgetstate command” on page 431
Displays the state of the GPFS daemon on one or more nodes.

Nodeclasses/{nodeclassName}: PUT

Change member nodes of a specific node class.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `nodeclass` request changes the member nodes of a specific node class. For more information about the fields in the data structures that are returned, see the topics “mmcrnodeclass command” on page 333, “mmdelnodeclass command” on page 379, “mmchnodeclass command” on page 251, and “mmlsnodeclass command” on page 520.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nodeclass/nodeclassName
```

where

nodeclasses/nodeclassName
Specifies the node class to be modified. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 145. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
nodeclassName	The name of the node class to be deleted. You can also specify keywords such as :all:, :all_local:, or :all_remote:	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
  "operation": "Operation",
  "members": "['node1', 'node2',..]"
}
```

"operation": "*add | delete | replace* "
The change operation to be performed.

"members": "[*List of nodes*]"
The list of nodes that are part of the node class.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
        ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example modifies the member nodes in the node class *cesNodes*.

Request URL:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json' -d "{
  "operation": "add",
  "members": "['mari-11', 'CloudOne']" }"
'https://198.51.100.1:443/scalemgmt/v2/nodeclasses/cesNodes'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": "[''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": "[''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": "[''EFSSG0740C There are not enough resources available to create
                   a new independent file set.'',  ...]",
        "stdout": "[''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "PUT",
        "url": "/scalemgmt/v2/nodeclasses/cesNodes",
         "data": "{
          "operation": "add",
           "members": "['mari-11', 'CloudOne']"
      },
       },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
```

```
      "status": "COMPLETED"
    }
  ]
}
```

“mmchnode command” on page 244
Changes node attributes.

“mmgetstate command” on page 431
Displays the state of the GPFS daemon on one or more nodes.

Nodes: GET

Gets information about nodes in the cluster.

Availability

Available on all IBM Spectrum Scale editions.

Description

The GET nodes request gets information about all nodes in the cluster. For more information about the fields in the data structures that are returned, see the topics “mmchnode command” on page 244 and “mmgetstate command” on page 431.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nodes
```

where

nodes
Specifies nodes as the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request data

No request data.

Response data

```
{
  "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "paging":
    {
    "next": "URL"
    },
   nodes: [
      {
        "adminNodename":"IP or host name",
        "nodeNumber":"Node ID",
        "config":
            {
             "adminLoginName":"Admin login name",
             "designatedLicense":"Designated license",
             "requiredLicense":"Required license",
            },
        "status":
           {
             "osName":"Operating system",
             "nodeState":"Health status",
             "gpfsState":"GPFS status",
             "productVersion":"Version",
            },
        "network":
            {
             "adminIPAddress":"IP address or host name",
             "daemonNodeName":"GPFS daemon node name",
             "daemonIPAddress":"IP address",
             "getcnfsNodeName":"Host name used by cNFS",
```

```
            },
        "roles"
            {
             "snmpNode":"yes | no",
             "managerNode":"yes | no",
             "gatewayNode":"yes | no",
             "cnfsNode":"yes | no",
             "cesNode":"yes | no",
             "quorumNode":"yes | no",
             "cloudGatewayNode":"yes | no",
             "otherNodeRoles":"Roles",
             "designation":"quorum | quorumManager |manager",
            },
        "cnfsInfo"
           {
             "cnfsState":"cNFS state",
             "cnfsGroupId":"cNFS gruop ID",
             "cnfsIplist":"cNFS IP aadresses",
           },
        "cesInfo":
            {
              ""cesState"":"CES state",
              "cesGroup":"CES group",
              "cesIpList":"CES IPs",
              "ipAddress":"IP address of this node",
            }
          }
        ]
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"nodes":

adminNodename":"*IP or host name*"
The host name that is used by the GPFS admin network.

"nodeNumber":"*Node ID*"
The GPFS node ID.

"config":

"adminLoginName":"*Admin login name*"
The name of the admin login.

"designatedLicense":"*Designated license*"
The license this node is running on.

"requiredLicense":"client | FPO | server"
Controls the type of GPFS required license that is associated with the nodes in the cluster.

"status":

"osName":"*Operating system*"
The name of Operating System running on this node.

"nodeState":"*Health status*"
The state of the node as reported by the **mmhealth node show** command.

"gpfsState":"*GPFS status*"
The state of GPFS on this node as reported the **mmhealth node show** command.

"productVersion":"*Version*"
The IBM Spectrum Scale version that is installed on this node.

"network":

"adminIPAddress":"*IP address or host name*"
The IP address that is used by the GPFS admin network.

"daemonNodeName":"*GPFS daemon node name*"
The host name that is used by the GPFS daemon network.

"daemonIPAddress":"*IP address*"
The IP address that is used by the GPFS daemon network.

"getcnfsNodeName":"*Host name used by cNFS*"
The hostname that is used by cNFS.

"roles":

"snmpNode":"yes | no"
Specifies whether this node is an SNMP node.

"managerNode":"yes | no"
Specifies whether this node is a manager node.

"gatewayNode":"yes | no"
Specifies whether this node is a gateway node.

"cnfsNode":"yes | no"
Specifies whether this node is a cNFS node.

"cesNode":"yes | no"
Specifies whether this node is a CES node.

"quorumNode":"yes | no"
Specifies whether this node is a quorum node.

"cloudGatewayNode":"yes | no"
Specifies whether this node is a cloud gateway node.

"otherNodeRoles":"*Roles*"
Other roles of this node.

"designation":"quorum | quorumManager |manager"
The designated role.

"cnfsInfo":

"cnfsState":"*cNFS state*"

"cnfsGroupId":"*cNFS gruop ID*"

"cnfsIplist":"*cNFS IP aadresses*"

"cesInfo":

""cesState"":"*CES state*"
The state of CNFS on this node.

"cesGroup":"*CES group*"
The CNFS group ID.

"cesIpList":"*CES IPs*"
A list of CNFS IP addresses.

"ipAddress":"*IP address of this node*"
The actual IP address of this node.

Examples

The following example gets information about the nodes.

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/nodes'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "nodes" : [ {
    "adminNodeName" : "mari-11.localnet.com"
  }, {
    "adminNodeName" : "mari-12.localnet.com"
  }, {
    "adminNodeName" : "mari-13.localnet.com"
  }, {
    "adminNodeName" : "mari-14.localnet.com"
  }, {
    "adminNodeName" : "mari-15.localnet.com"
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully"
  }
}
```

Using the field parameter ":all:" returns entire details of the nodes that are available in the cluster. For example:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/nodes?fields=:all:'
```

```
{
  "nodes" : [ {
    "nodeNumber" : 1,
    "adminNodeName" : "mari-11.localnet.com",
    "config" : {
      "adminLoginName" : "root",
      "designatedLicense" : "server",
      "requiredLicense" : "server"
    },
    "status" : {
      "gpfsState" : "HEALTHY",
      "nodeState" : "HEALTHY",
      "osName" : "Red Hat Enterprise Linux Server 7.2 (Maipo)",
      "productVersion" : "4.2.3.0"
    },
    "network" : {
      "adminIPAddress" : "198.52.100.11",
      "daemonIPAddress" : "198.52.100.11",
      "daemonNodeName" : "mari-11.localnet.com"
    },
    "roles" : {
      "cesNode" : false,

      "cloudGatewayNode" : false,
      "cnfsNode" : false,
      "designation" : "quorum",
      "gatewayNode" : false,
      "managerNode" : false,
      "otherNodeRoles" : "perfmonNode",
      "quorumNode" : true,
      "snmpNode" : false
    }
  }, {
    "nodeNumber" : 2,
    "adminNodeName" : "mari-12.localnet.com",
    "config" : {
      "adminLoginName" : "root",
      "designatedLicense" : "server",
      "requiredLicense" : "server"
    },
    "status" : {
```

```
      "gpfsState" : "HEALTHY",
      "nodeState" : "HEALTHY",
      "osName" : "Red Hat Enterprise Linux Server 7.2 (Maipo)",
      "productVersion" : "4.2.3.0"
    },
    "network" : {
      "adminIPAddress" : "198.52.100.12",
      "daemonIPAddress" : "198.52.100.12",
      "daemonNodeName" : "mari-12.localnet.com"
    },
    "roles" : {
      "cesNode" : true,

      "cloudGatewayNode" : false,
      "cnfsNode" : false,
      "designation" : "quorum",
      "gatewayNode" : false,
      "managerNode" : false,
      "otherNodeRoles" : "perfmonNode,cesNode,cloudNodeMarker",
      "quorumNode" : true,
      "snmpNode" : false
    },
    "cesInfo" : {
      "cesGroup" : "",
      "cesIpList" : "198.51.100.43,198.51.100.22,198.51.100.28,198.51.100.34,198.51.100.40,
                     198.51.100.46,198.51.100.19,198.51.100.25,198.51.100.31,198.51.100.37",
      "cesState" : "e",
      "ipAddress" : "198.52.100.12"
    }
  }, {
    "nodeNumber" : 3,
    "adminNodeName" : "mari-13.localnet.com",
    "config" : {
      "adminLoginName" : "root",
      "designatedLicense" : "server",
      "requiredLicense" : "server"
    },
    "status" : {
      "gpfsState" : "HEALTHY",
      "nodeState" : "HEALTHY",
      "osName" : "Red Hat Enterprise Linux Server 7.2 (Maipo)",
      "productVersion" : "4.2.3.0"
    },
    "network" : {
      "adminIPAddress" : "198.52.100.13",
      "daemonIPAddress" : "198.52.100.13",
      "daemonNodeName" : "mari-13.localnet.com"
    },
    "roles" : {
      "cesNode" : true,

      "cloudGatewayNode" : false,
      "cnfsNode" : false,
      "designation" : "quorum",
      "gatewayNode" : false,
      "managerNode" : false,
      "otherNodeRoles" : "perfmonNode,cesNode,cloudNodeMarker",
      "quorumNode" : true,
      "snmpNode" : false
    },
    "cesInfo" : {
      "cesGroup" : "",
      "cesIpList" : "198.51.100.41,198.51.100.47,198.51.100.20,198.51.100.26,198.51.100.32,
                     198.51.100.38,198.51.100.44,198.51.100.23,198.51.100.29,198.51.100.35",
      "cesState" : "e",
      "ipAddress" : "198.52.100.13"
    }
  }, {
    "nodeNumber" : 4,
    "adminNodeName" : "mari-14.localnet.com",
    "config" : {
      "adminLoginName" : "root",
      "designatedLicense" : "server",
      "requiredLicense" : "server"
    },
    "status" : {
      "gpfsState" : "HEALTHY",
      "nodeState" : "HEALTHY",
      "osName" : "Red Hat Enterprise Linux Server 7.2 (Maipo)",
      "productVersion" : "4.2.3.0"
    },
    "network" : {
```

```
      "adminIPAddress" : "198.52.100.14",
      "daemonIPAddress" : "198.52.100.14",
      "daemonNodeName" : "mari-14.localnet.com"
    },
    "roles" : {
      "cesNode" : true,

      "cloudGatewayNode" : false,
      "cnfsNode" : false,
      "designation" : "client",
      "gatewayNode" : false,
      "managerNode" : false,
      "otherNodeRoles" : "perfmonNode,cesNode,cloudNodeMarker",
      "quorumNode" : false,
      "snmpNode" : false
    },
    "cesInfo" : {
      "cesGroup" : "",
      "cesIpList" : "198.51.100.45,198.51.100.24,198.51.100.30,198.51.100.36,198.51.100.42,
                     198.51.100.48,198.51.100.21,198.51.100.27,198.51.100.33,198.51.100.39",
      "cesState" : "e",
      "ipAddress" : "198.52.100.14"
    }
  }, {
    "nodeNumber" : 5,
    "adminNodeName" : "mari-15.localnet.com",
    "config" : {
      "adminLoginName" : "root",
      "designatedLicense" : "server",
      "requiredLicense" : "server"
    },
    "status" : {
      "gpfsState" : "HEALTHY",
      "nodeState" : "HEALTHY",
      "osName" : "Red Hat Enterprise Linux Server 7.2 (Maipo)",
      "productVersion" : "4.2.3.0"
    },
    "network" : {
      "adminIPAddress" : "198.52.100.15",
      "daemonIPAddress" : "198.52.100.15",
      "daemonNodeName" : "mari-15.localnet.com"
    },
    "roles" : {
      "cesNode" : true,

      "cloudGatewayNode" : false,
      "cnfsNode" : false,
      "designation" : "client",
      "gatewayNode" : false,
      "managerNode" : false,
      "otherNodeRoles" : "perfmonNode,cesNode,cloudNodeMarker",
      "quorumNode" : false,
      "snmpNode" : false
    },
    "cesInfo" : {
      "cesGroup" : "",
      "cesIpList" : "198.51.100.10,198.51.100.14,198.51.100.12,198.51.100.18,198.51.100.16,
                     198.51.100.49,198.51.100.11,198.51.100.15,198.51.100.13,198.51.100.17",
      "cesState" : "e",
      "ipAddress" : "198.52.100.15"
    }
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully"
  }
}
```

“mmchnode command” on page 244
Changes node attributes.

“mmgetstate command” on page 431
Displays the state of the GPFS daemon on one or more nodes.

Nodes: POST

Add one or more nodes to the IBM Spectrum Scale cluster.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST nodes request adds one or more nodes to the cluster. For more information about the fields in the data structures that are returned, see the topics “mmaddnode command” on page 34.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nodes
```

where

nodes
Specifies nodes as the target of the operation. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 146. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
  {
          "nodesDesc": "Descriptions",
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"nodesDesc":"*Descriptions*"
Descriptions of the nodes to be added to the cluster.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
```

```
},
jobs: [
   {
      "result":"",
        {
          "commands":"String",
          "progress":"String,
          "exitCode":"Exit code",
          "stderr":"Error",
          "stdout":"String",
        },
      "request":" ",
        {
          "type":"{GET | POST | PUT | DELETE}",
          "url":"URL",
          "data":"",
        }
      "jobId":"ID",
      "submitted":"Time",
      "completed":Time",
      "status":"Job status",
      }
     ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following API command adds two nodes to the cluster.

Request URL:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
-d '{"nodesDesc": ['mari-16:manager-quorum', 'mari-17::mari-17_admin']}'
'https://198.51.100.1:443/scalemgmt/v2/nodes'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": "[''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": "[''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": "[''EFSSG0740C There are not enough resources available to create
                  a new independent file set.'',  ...]",
        "stdout": "[''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "POST",
        "url": "/scalemgmt/v2/nodeclasses",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

“mmaddnode command” on page 34
Adds nodes to a GPFS cluster.

Nodes/afm/mapping: GET

Lists all node mappings for AFM and cloud object storage (COS).

Availability

Available on all IBM Spectrum Scale editions.

Description

The GET `nodes/afm/mapping` request lists all node mappings for AFM and cloud object storage (COS). For more information about the fields in the data structures that are returned, see "mmafmconfig command" on page 45.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nodes/afm/mapping
```

where

nodes/afm/mapping
Specifies the target of the request. Required.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 147. List of parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   },
 "paging":
  {
    "next": "URL",
```

```
    "fields": "Fields",
    "filter": "Filters",
    "baseUrl": "Base URL",
    "lastId": "ID of the last element",
  },
   mappings: [
      {
      "mapName": Map name,
      "exportMap": "Export server or gateway node map details",
      ],
     }
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging":
Paging details.

"next": "*URL*",
The URL to retrieve the next page. Paging is enabled when more than 1000 objects would be returned by the query.

"fields": "*Fields*",
The fields used in the original request.

"filter": "*Filters*",
The filter used in the original request.

"baseUrl": "*Base URL*",
The URL of the request without any parameters.

"lastId": "*ID of the last element*",
The ID of the last element that can be used to retrieve the next elements.

"mappings":
An array of elements that describe one mapping.

"mapName" : "*Map name*"
Name of the mapping.

"exportMap" : "*Export server and gateway node map details*"
Lists the export server and gateway node maps.

Examples

The following example lists all node mapping for AFM and COS.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/node/afm/mapping'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": 200,
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/filesystems/gpfs0/filesets?lastId=10001",
    "fields": "period,restrict,sensorName",
    "filter": "usedInodes>100,maxInodes>1024",
    "baseUrl": "/scalemgmt/v2/nodes/afm/mapping",
    "lastId": 10001
  },
   "mappings": [
    {
      "mapName": "myMap",
      "exportMap": "[ testnode-11/node-13, testnode-11/node-14]"
    }
  ]
}
```

Related reference

“mmafmconfig command” on page 45
Can be used to manage home caching behavior and mapping of gateways and home NFS exported servers.

Nodes/afm/mapping: POST

Creates node mapping for AFM and Cloud Object Store.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `Nodes/afm/mapping` request creates node mapping for AFM and Cloud Object Storage. For more information about the fields in the data structures that are returned, see "mmafmcosconfig command" on page 50.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nodes/afm/mapping
```

where

/nodes/afm/mapping
Specifies the target of the request.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 148. List of parameters

Parameter name	Description and applicable keywords	Required/optional
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
      "mapName": Map name,
      "exportMap": "Export server or gateway node map details",
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"mapName" : "*Map name*"
Name of the mapping.

"exportMap" : "*Export server and gateway node map details*"
Lists the export server and gateway node maps.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero value denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to create a node mapping for AFM and Cloud Object Storage.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
-d '{ \
   "mapName": "myMap", \
   "exportMap": "[ testnode-11/node-13, testnode-11/node-14]" \
 }' 'https://198.51.100.1:443/scalemgmt/v2/node/afm/mapping/'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": ["''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": ["''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": ["''EFSSG0740C There are not enough resources available to create
                  a new independent file set.'',  ...]",
        "stdout": ["''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "POST",
        "url": "/scalemgmt/v2/node/afm/mapping",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

Related reference

“mmafmcosconfig command” on page 50
Creates and displays an AFM to cloud object storage fileset.

Nodes/afm/mapping: DELETE

Deletes a node mapping for AFM and Cloud Object Storage.

Availability

Available on all IBM Spectrum Scale editions.

Description

The DELETE nodes/afm/mapping*{mappingName}* request deletes a node mapping for AFM and Cloud Object Storage. For more information about the fields in the data structures that are returned, see "mmafmconfig command" on page 45.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nodes/afm/
mapping{mappingName}
```

where

nodes/afm/mapping
Specifies the target of the request.

{mappingName}
Specifies the map to be deleted.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 149. List of parameters

Parameter name	Description and applicable keywords	Required/optional
mappingName	Name of the node mapping to be deleted.	Required

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
```

```
            "stderr":"Error",
            "stdout":"String",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted":"Time",
        "completed":Time",
        "status":"Job status",
        }
      ],
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero value denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example deletes the map myMap.

Request data:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
   'https://198.51.100.1:443/scalemgmt/v2/node/afm/mapping/myMap'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": "[''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": "[''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": "[''EFSSG0740C There are not enough resources available to create
                  a new independent file set.'',  ...]",
        "stdout": "[''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "DELETE",
        "url": "/scalemgmt/v2/node/afm/mapping/myMap",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

Related reference

“mmafmconfig command” on page 45
Can be used to manage home caching behavior and mapping of gateways and home NFS exported servers.

Nodes/afm/mapping/{mappingName}: GET

Lists details of a node mapping for AFM and Cloud Object Storage (COS).

Availability

Available on all IBM Spectrum Scale editions.

Description

The GET `nodes/afm/mapping/{mappingName}` request lists all node mappings for AFM and Cloud Object Storage. For more information about the fields in the data structures that are returned, see "mmafmconfig command" on page 45.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nodes/afm/mapping/
{mappingName}
```

where

nodes/afm/mapping
Specifies the target of the request.

mappingName
Name of the map.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 150. List of parameters

Parameter name	Description and applicable keywords	Required/optional
mappingName	Node mapping name.	Required
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   },
 "paging":
  {
    "next": "URL",
```

```
    "fields": "Fields",
    "filter": "Filters",
    "baseUrl": "Base URL",
    "lastId": "ID of the last element",
  },
   mappings: [
      {
      "mapName": Map name,
      "exportMap": "Export server or gateway node map details",
      }
      ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging":
Paging details.

"next": "*URL*",
The URL to retrieve the next page. Paging is enabled when more than 1000 objects would be returned by the query.

"fields": "*Fields*",
The fields used in the original request.

"filter": "*Filters*",
The filter used in the original request.

"baseUrl": "*Base URL*",
The URL of the request without any parameters.

"lastId": "*ID of the last element*",
The ID of the last element that can be used to retrieve the next elements.

"mappings":
An array of elements that describe one mapping.

"mapName" : "*Map name*"
Name of the mapping.

"exportMap" : "*Export server and gateway node map details*"
Lists the export server and gateway node maps.

Examples

The following example gets the details of the node mapping for AFM and Cloud Object Storage.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/node/afm/mapping/myMap'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": 200,
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/filesystems/gpfs0/filesets?lastId=10001",
    "fields": "period,restrict,sensorName",
    "filter": "usedInodes>100,maxInodes>1024",
    "baseUrl": "/scalemgmt/v2/nodes/afm/mapping/myMap",
    "lastId": 10001
  },
   "mappings": [
    {
      "mapName": "myMap",
      "exportMap": "[ testnode-11/node-13, testnode-11/node-14]"
    }
  ]
}
```

Related reference

“mmafmconfig command” on page 45
Can be used to manage home caching behavior and mapping of gateways and home NFS exported servers.

Nodes/afm/mapping/{mappingName}: PUT

Changes a node mapping for AFM and Cloud Object Storage.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `nodes/afm/mapping/{mappingName}` request changes a node mapping for AFM and Cloud Object Storage. For more information about the fields in the data structures that are returned, see "mmafmconfig command" on page 45.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nodes/afm/mapping/
{mappingName}
```

where

nodes/afm/mapping
Specifies the target of the request.

{mappingName}
Specifies the map to be modified.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 151. List of parameters

Parameter name	Description and applicable keywords	Required/optional
mappingName	Node mapping name.	Required
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
     "exportMap": "Export server or gateway node map details",
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"exportMap" : "*Export server and gateway node map details*"
Lists the export server and gateway node maps.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success and a nonzero value denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to change the node mapping myMap:

Request data:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
-d '{ \
    "exportMap": "[ testnode-11/node-13, testnode-11/node-14]" \
 }' 'https://198.51.100.1:443/scalemgmt/v2/node/afm/mapping/myMap'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": "[''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": "[''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": "[''EFSSG0740C There are not enough resources available to create
                   a new independent file set.'',  ...]",
        "stdout": "[''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "PUT",
        "url": "/scalemgmt/v2/node/afm/mapping/myMap",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

Related reference

“mmafmconfig command” on page 45
Can be used to manage home caching behavior and mapping of gateways and home NFS exported servers.

Nodes/{name}: DELETE

Delete a specific node or all nodes of a node class from the IBM Spectrum Scale cluster.

Availability

Available on all IBM Spectrum Scale editions.

Description

The DELETE nodes/name request deletes one or more nodes from the cluster. For more information about the fields in the data structures that are returned, see the topics “mmdelnode command” on page 375.

Prerequisites

Ensure the following before you run this API command:

- GPFS needs to be stopped on all affected nodes.
- Affected nodes cannot be used as NSD server.
- CES must be disabled on all affected nodes.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nodes/name
```

where

nodes/name
Specifies the name of the node or all nodes in a node class to be deleted. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 152. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
name	Name of the node or node class. If node class name is specified, all nodes of the node class will be deleted.	Required.

Request data

```
  {
        "name": "Name",
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"name":"*Name of node or node class*"
Specify the nodes to be deleted.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*'
Array of commands that are run in this job.

"progress":"*String*'
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following API command deletes the node *node1* from the cluster.

Request URL:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/nodes/node1'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": "[''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": "[''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": "[''EFSSG0740C There are not enough resources available to create
                  a new independent file set.'',  ...]",
        "stdout": "[''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "DELETE",
        "url": "/scalemgmt/v2/nodes/node1",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
```

```
  ]
}
```

“mmdelnode command” on page 375
Removes one or more nodes from a GPFS cluster.

Nodes/{name}: GET

Gets information about a node.

Availability

Available on all IBM Spectrum Scale editions.

Description

The GET nodes/nodeName request gets information about the specified node. For more information about the fields in the data structures that are returned, see the topics “mmchnode command” on page 244 and “mmgetstate command” on page 431.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nodes/nodeName
```

where

nodes
Is the nodes resource. Required.

nodeName
Specifies the node about which you want to get information. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request data

No request data.

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 153. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.
name	Name of the node.	Required.

Response data

```
{
  "status": {
      "code":ReturnCode",
```

```
      "message":"ReturnMessage"
   },
   "paging":
    {
    "next": "URL"
    },
   nodes: [
      {
        "adminNodename":"IP or host name",
        "nodeNumber":"Node ID",
        "config":
            {
             "adminLoginName":"Admin login name",
             "designatedLicense":"Designated license",
             "requiredLicense":"Required license",
            },
        "status":
           {
             "osName":"Operating system",
             "nodeState":"Health status",
             "gpfsState":"GPFS status",
             "productVersion":"Version",
            },
        "network":
            {
             "adminIPAddress":"IP address or host name",
             "daemonNodeName":"GPFS daemon node name",
             "daemonIPAddress":"IP address",
             "getcnfsNodeName":"Host name used by cNFS",
            },
        "roles"
            {
             "snmpNode":"yes | no",
             "managerNode":"yes | no",
             "gatewayNode":"yes | no",
             "cnfsNode":"yes | no",
             "cesNode":"yes | no",
             "quorumNode":"yes | no",
             "cloudGatewayNode":"yes | no",
             "otherNodeRoles":"Roles",
             "designation":"quorum | quorumManager |manager",
            },
        "cnfsInfo"
           {
             "cnfsState":"cNFS state",
             "cnfsGroupId":"cNFS gruop ID",
             "cnfsIplist":"cNFS IP aadresses",
           },
        "cesInfo":
            {
              ""cesState"":"CES state",
              "cesGroup":"CES group",
              "cesIpList":"CES IPs",
              "ipAddress":"IP address of this node",
            }
          }
        ]
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"nodes":

adminNodename":"*IP or host name*"
The host name that is used by the GPFS admin network.

"nodeNumber":"*Node ID*"
The GPFS node ID.

"config":

"adminLoginName":"*Admin login name*"
The name of the admin login.

"designatedLicense":"*Designated license*"
The license this node is running on.

"requiredLicense":"client | FPO | server"
Controls the type of GPFS required license that is associated with the nodes in the cluster.

"status":

"osName":"*Operating system*"
The name of Operating System running on this node.

"nodeState":"*Health status*"
The state of the node as reported by the **mmhealth node show** command.

"gpfsState":"*GPFS status*"
The state of GPFS on this node as reported the **mmhealth node show** command.

"productVersion":"*Version*"
The IBM Spectrum Scale version that is installed on this node.

"network":

"adminIPAddress":"*IP address or host name*"
The IP address that is used by the GPFS admin network.

"daemonNodeName":"*GPFS daemon node name*"
The host name that is used by the GPFS daemon network.

"daemonIPAddress":"*IP address*"
The IP address that is used by the GPFS daemon network.

"getcnfsNodeName":"*Host name used by cNFS*"
The hostname that is used by cNFS.

"roles":

"snmpNode":"yes | no"
Specifies whether this node is an SNMP node.

"managerNode":"yes | no"
Specifies whether this node is a manager node.

"gatewayNode":"yes | no"
Specifies whether this node is a gateway node.

"cnfsNode":"yes | no"
Specifies whether this node is a cNFS node.

"cesNode":"yes | no"
Specifies whether this node is a CES node.

"quorumNode":"yes | no"
Specifies whether this node is a quorum node.

"cloudGatewayNode":"yes | no"
Specifies whether this node is a cloud gateway node.

"otherNodeRoles":"*Roles*"
Other roles of this node.

"designation":"quorum | quorumManager |manager"
The designated role.

"cnfsInfo":

"cnfsState":"*cNFS state*"

"cnfsGroupId":"*cNFS gruop ID*"

"cnfsIplist":"*cNFS IP aadresses*"

"cesInfo":

""cesState"":"*CES state*"
The state of CNFS on this node.

"cesGroup":"*CES group*"
The CNFS group ID.

"cesIpList":"*CES IPs*"
A list of CNFS IP addresses.

"ipAddress":"*IP address of this node*"
The actual IP address of this node.

Examples

The following example gets information about the node `testnode1-d.localnet.com`.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/node/testnode1-d.localnet.com
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/node/testnode1-d.localnet.com?lastId=1001"
  },
  "nodes": [
    {
      "adminNodeName": "testnode1-a.localnet.com",
      "nodeNumber": "1",
      "config": {
        "adminLoginName": "root",
        "designatedLicense": "server",
        "requiredLicense": "true"
      },
      "status": {
        "osName": "Red Hat Enterprise Linux Server 7.2 (Maipo)",
        "nodeState": "HEALTHY",
        "gpfsState": "HEALTHY",
        "productVersion": "4.2.3.0"
      },
      "network": {
        "adminIPAddress": "10.0.200.21",
        "daemonNodeName": "testnode1-d.localnet.com",
        "daemonIPAddress": "10.0.100.21",
        "getcnfsNodeName": "string"
      },
      "roles": {
        "snmpNode": "true",
        "managerNode": "false",
        "gatewayNode": "false",
        "cnfsNode": "true",

        "cesNode": "false",
        "quorumNode": "true",
        "cloudGatewayNode": "true",
        "otherNodeRoles": "perfmonNode,cesNode",
```

```
        "designation": "quorum"
      },
      "cnfsInfo": {
        "cnfsState": "enabled",
        "cnfsGroupId": "5",
        "cnfsIplist": "10.0.100.1,10.0.100.2"
      },
      "cesInfo": {
        "cesState": "e",
        "cesGroup": "LAN_4",
        "cesIpList": "192.168.2.11,192.168.2.13,192.168.2.15",
        "ipAddress": "10.0.100.23"
      }
    }
  ]
}
```


“mmchnode command” on page 244
Changes node attributes.

“mmgetstate command” on page 431
Displays the state of the GPFS daemon on one or more nodes.

Nodes/{name}: PUT

Enables or disables quorum and gateway designations of a node or node class.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT nodes/*{name}* request enables or disables quorum and gateway designations of a node or node class. For more information about the fields in the data structures that are returned, see “mmchnode command” on page 244.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nodes/{name}
```

where

nodes/*{name}*
: Specifies the target of the request. Required.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 154. List of parameters

Parameter name	Description and applicable keywords	Required/optional
{name}	Node or node class name.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
  "quorum": "true | false",
  "gateway": "true | false",
 }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"quorum": "*true | false*"
: Set to `true` if you want to enable quorum. Set to `false` to disable it. Omit this parameter if you do not want to change quorum.

"gateway": "*true | false*"
Set to `true` if you want to designate as a gateway. Set to `false` to disable an existing designation as a gateway. Omit this parameter if you do not want to change the gateway designation.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | PUT | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
        ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to change quorum and gateway designation of a node:

Request data:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json' -d '{ \
   "quorum": true, \
   "gateway": true \
 }' 'https://198.51.100.1:443/scalemgmt/v2/nodes/scale-node'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": ["''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": ["''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": ["''EFSSG0740C There are not enough resources available to create
                   a new independent file set.'',  ...]",
        "stdout": ["''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "PUT",
        "url": "/scalemgmt/v2/nodes/scale-node",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

Related reference

“mmchnode command” on page 244
Changes node attributes.

Nodes/{name}/health/events: GET

Gets details about system health events that are reported in a node.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET nodes/nodeName/health/events` request gets information about the system health events that are reported in the specified node. For more information about the fields in the data structures that are returned, see the topics “mmhealth command” on page 441 and “mmces command” on page 133.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nodes/nodeName/health/events
```

where

nodes/nodesName
Specifies the node about which you want to get information. Required.

health/events
Specifies that you need to get the details of the system health events reported on the node. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 155. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.
name	Name of the node.	Required.

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode",
```

```
      "message":"ReturnMessage"
     },
    "paging":
     {
     "next": "URL"
     },
   events: [
      {
         "oid":"ID",
         "component":"Component",
         "reportingnode":" Node Name",
         "type":" {TIP | STATE_CHANGE | NOTICE}",
         "activeSince":"Time",
         "name":"Event Name",
         "message":"Message",
         "entityType":"Entity Type",
         "entityName":" Name",
         "userAction":" Action",
         "description":" Description",
         "state":"Event State",
         "severity":"Severity",
         "parentName":"Name",
      }
   ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"events":
An array of elements that describe the system health event. Each element describes one node.

"oid":"*ID*
The internal unique ID of the event.

"component":"*Component*
The component to which the event belongs.

"reportingnode":" *Node Name*
The node in which the event is reported.

"type":"{TIP | STATE_CHANGE | NOTICE}
Specifies the event type.

"activeSince":"*Time*
The time at which the event occurred.

"name":"*Event Name*
The name of the event.

"message":"*Message*
The event message.

"entityType":"*Entity Type*
The type of the entity for which the event occurred.

"entityName":" *Name*
The name of the entity for which the event occurred.

"userAction":" *Action*
The user action that is required to resolve the event.

"description":" ***Description***
Description of the event.

"state":"*Event State*
The state of the event.

"severity":"*Severiy*
Severity of the event.

"parentName":"*Name*
Name of the parent event to which this event belongs.

Examples

The following example gets information about the events reported in the node `testnode1-d.localnet.com`

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/node/testnode1-d.localnet.com/health/events
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.


```
{
  "events" : [ {
    "activeSince" : "2017-03-13 10:55:15,809",
    "component" : "FILESYSTEM",
    "description" : "The inode usage in the fileset reached a normal level.",
    "entityName" : "fset1",
    "entityType" : "FILESET",
    "message" : "The inode usage of fileset fset1 in file system gpfs0 reached a normal level.",
    "name" : "inode_normal",
    "oid" : 690,
    "parentName" : "gpfs0",
    "reportingNode" : "mari-11.localnet.com",
    "severity" : "INFO",
    "state" : "HEALTHY",
    "type" : "STATE_CHANGE",
    "userAction" : "N/A"
  }, {
    "activeSince" : "2017-03-13 11:05:15,912",
    "component" : "FILESYSTEM",
    "description" : "The inode usage in the fileset reached a normal level.",
    "entityName" : "fset2",
    "entityType" : "FILESET",
    "message" : "The inode usage of fileset fset2 in file system gpfs0 reached a normal level.",
    "name" : "inode_normal",
    "oid" : 691,
    "parentName" : "gpfs0",
    "reportingNode" : "mari-11.localnet.com",
    "severity" : "INFO",
    "state" : "HEALTHY",
    "type" : "STATE_CHANGE",
    "userAction" : "N/A"
  }, {
    "activeSince" : "2017-03-15 18:44:42,568",
    "component" : "GUI",
    "description" : "The GUI service is running",
    "entityName" : "mari-11.localnet.com",
    "entityType" : "NODE",
    "message" : "GUI service as expected, state is started",
    "name" : "gui_up",
    "oid" : 882,
    "reportingNode" : "mari-11.localnet.com",
    "severity" : "INFO",
    "state" : "HEALTHY",
    "type" : "STATE_CHANGE",
    "userAction" : "N/A"
  }, {
    "activeSince" : "2017-03-16 07:07:06,418",
```

```
    "component" : "GUI",
    "description" : "The GUI checks the reachability of all nodes.",
    "entityName" : "mari-11.localnet.com",
    "entityType" : "NODE",
    "message" : "The GUI can reach the node mari-12.localnet.com",
    "name" : "gui_reachable_node",
    "oid" : 884,
    "reportingNode" : "mari-11.localnet.com",
    "severity" : "INFO",
    "state" : "HEALTHY",
    "type" : "STATE_CHANGE",
    "userAction" : "None needed."
  }
  ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully"
  }
}
```

“mmhealth command” on page 441
Monitors health status of nodes.

“mmces command” on page 133
Manage CES (Cluster Export Services) configuration.

Nodes/{name}/health/states: GET

Gets details about system health states for the node or nodes of node class.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET nodes/nodeName/health/states` request gets information about the system health states of the specified node or node class. For more information about the fields in the data structures that are returned, see the topics “mmhealth command” on page 441 and “mmces command” on page 133.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nodes/nodeName/health/states
```

where

nodes/nodeName
Specifies the node about which you want to get information. Required.

health/states
Specifies that you need to get the details of the system health state of the node. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 156. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.
name	Name of the node.	Required.

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode",
```

```
      "message":"ReturnMessage"
     },
    "paging":
     {
     "next": "URL"
     },
   events: [
      {
         "oid":"ID",
         "component":"Component",
         "reportingnode":" Node Name",
         "type":" {TIP | STATE_CHANGE | NOTICE}",
         "activeSince":"Time",
         "name":"Event Name",
         "message":"Message",
         "entityType":"Entity Type",
         "entityName":" Name",
         "userAction":" Action",
         "description":" Description",
         "state":"Event State",
         "severity":"Severity",
         "parentName":"Name",
      }
   ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"events":
An array of elements that describe the system health event. Each element describes one node.

"oid":"*ID*
The internal unique ID of the event.

"component":"*Component*
The component to which the event belongs.

"reportingnode":" *Node Name*
The node in which the event is reported.

"type":"{TIP | STATE_CHANGE | NOTICE}
Specifies the event type.

"activeSince":"*Time*
The time at which the event occurred.

"name":"*Event Name*
The name of the event.

"message":"*Message*
The event message.

"entityType":"*Entity Type*
The type of the entity for which the event occurred.

"entityName":" *Name*
The name of the entity for which the event occurred.

"userAction":" *Action*
The user action that is required to resolve the event.

"description":" ***Description***
Description of the event.

"state":"*Event State*
The state of the event.

"severity":"*Severiy*
Severity of the event.

"parentName":"*Name*
Name of the parent event to which this event belongs.

Examples

The following example gets information about the events reported in the node `testnode1-d.localnet.com`.

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/node/testnode1-d.localnet.com/health/states
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "states" : [ {
    "activeSince" : "2017-03-02 15:21:57,431",
    "component" : "THRESHOLD",
    "entityName" : "mari-11.localnet.com",
    "entityType" : "NODE",
    "oid" : 4,
    "reportingNode" : "mari-11.localnet.com",
    "state" : "HEALTHY"
  }, {
    "activeSince" : "2017-03-02 15:21:58,284",
    "component" : "NETWORK",
    "entityName" : "mari-11.localnet.com",
    "entityType" : "NODE",
    "oid" : 5,
    "reportingNode" : "mari-11.localnet.com",
    "state" : "HEALTHY"
  }, {
    "activeSince" : "2017-03-02 15:24:57,143",
    "component" : "PERFMON",
    "entityName" : "mari-11.localnet.com",
    "entityType" : "NODE",
    "oid" : 6,
    "reportingNode" : "mari-11.localnet.com",
    "state" : "HEALTHY"
  }, {
    "activeSince" : "2017-03-02 15:21:58,284",
    "component" : "NETWORK",
    "entityName" : "eth0",
    "entityType" : "NIC",
    "oid" : 9,
    "parentName" : "mari-11.localnet.com",
    "reportingNode" : "mari-11.localnet.com",
    "state" : "HEALTHY"
  }
  ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully"
  }
}
```

“mmhealth command” on page 441
Monitors health status of nodes.

“mmces command” on page 133

Manage CES (Cluster Export Services) configuration.

Nodes/{name}/services: GET

Get status of all services that are hosted on a node or node class in the cluster.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET nodes/{name}/services` request gets information about the services that are configured on a specific node or node class. For more information about the fields in the returned data structure, see “mmces command” on page 133.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nodes/name/services
```

where

nodes/name/services
Specifies services in a particular node or node class as the resource of the GET call.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 157. List of parameters

Parameter name	Description and applicable keywords	Required/optional
name	Name of the node or node class.	Required.
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Request data

No request data.

Response data

```
{
"status":
   {
      "code": ReturnCode
      "message": "ReturnMessage",
   },
"paging":
```

```
   {
      "next": Next page URL
      "fields": "Fields",
      "filter": Filter
      "baseUrl": "URL",
      "lastID": ID
      ,
   },
{
   "serviceStatus": [
      {
      "nodeName": Node name
      "serviceName": "Service name",
      "state": State
      "healthState": "Health state",

       }
}
```

"status":
Return status.

"code": *ReturnCode*,
The HTTP status code that was returned by the request.

"message": "*ReturnMessage*"
The return message.

"paging":
An array of information about the paging information that is used for displaying the details.

"next": "*Next page URL*"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects would be returned by the query.

"fields": "*Fields in the request*"
The fields that are used in the original request.

"filter": "*Filters used in the request*"
The filter that is used in the original request.

"baseUrl": "*URL*"
The URL of the request without any parameters.

"lastId": "*ID*"
The ID of the last element that can be used to retrieve the next elements.

"serviceStatus":
An array of information that provides the details of the services configured on the node.

"nodeName": *Node name*
The node where the service is hosted.

"serviceName": "*Service name*"
Name of the service.

"state": *State*
Status of the service.

"healthState": "*Health state*"
Health status of the service.

Examples

The following example gets information about the services that are configured on the node *mari-13.localnet.com*.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/nodes/mari-13.localnet.com/services'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

The **serviceStatus** array returns the details of the services that are configured on the node *mari-13.localnet.com*.

```
{
  "serviceStatus" : [ {
    "nodeName" : "mari-12.localnet.com",
    "serviceName" : "NFS",
    "state" : "running",
    "healthState" : "HEALTHY"
  }, {
    "nodeName" : "mari-12.localnet.com",
    "serviceName" : "SMB",
    "state" : "running",
    "healthState" : "HEALTHY"
  }, {
    "nodeName" : "mari-12.localnet.com",
    "serviceName" : "OBJ",
    "state" : "running",
    "healthState" : "HEALTHY"
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Related reference

“mmces command” on page 133
Manage CES (Cluster Export Services) configuration.

Nodes/{name}/services/{serviceName}: GET

Gets status of a specific service that is hosted on a node or node class in the cluster.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET nodes/name/services/serviceName` request gets information about a specific service that is hosted on a specific node or node class. For more information about the fields in the returned data structure, see "mmces command" on page 133.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nodes/name/services/
serviceName
```

where

nodes/name
Specifies the node or node class on which the service is hosted.

/services/serviceName
Specifies the service about which you need the details.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 158. List of parameters

Parameter name	Description and applicable keywords	Required/optional
name	Name of the node or node class on which the service is hosted.	Required.
serviceName	Name of the service about which the information is required.	Required.
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Request data

No request data.

Response data

```
{
"status":
   {
      "code": ReturnCode
      "message": "ReturnMessage",
   },
"paging":
   {
      "next": Next page URL
      "fields": "Fields",
      "filter": Filter
      "baseUrl": "URL",
      "lastID": ID
      ,
   },
{
   "serviceStatus": [
      {
      "nodeName": Node name
      "serviceName": "Service name",
      "state": State
      "healthState": "Health state",

       }
}
```

"status":
Return status.

"code": *ReturnCode*,
The HTTP status code that was returned by the request.

"message": "*ReturnMessage*"
The return message.

"paging":
An array of information about the paging information that is used for displaying the details.

"next": "*Next page URL*"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects would be returned by the query.

"fields": "*Fields in the request*"
The fields that are used in the original request.

"filter": "*Filters used in the request*"
The filter that is used in the original request.

"baseUrl": "*URL*"
The URL of the request without any parameters.

"lastId": "*ID*"
The ID of the last element that can be used to retrieve the next elements.

"serviceStatus":
An array of information that provides the details of the services configured on the node.

"nodeName": *Node name*
The node where the service is hosted.

"serviceName": "*Service name*"
Name of the service.

"state": *State*
Status of the service.

"healthState": "*Health state*"
Health status of the service.

Examples

Example 1:

The following example gets information about the SMB service on the CES nodes.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/nodes/cesNodes/services/smb'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

The **serviceStatus** array returns the details of the services that are configured on the node *scale01*.

```
{
  "serviceStatus" : [ {
    "nodeName" : "mari-14.localnet.com",
    "serviceName" : "SMB",
    "state" : "running",
    "healthState" : "HEALTHY"
  }, {
    "nodeName" : "mari-15.localnet.com",
    "serviceName" : "SMB",
    "state" : "running",
    "healthState" : "HEALTHY"
  }, {
    "nodeName" : "mari-13.localnet.com",
    "serviceName" : "SMB",
    "state" : "running",
    "healthState" : "HEALTHY"
  }, {
    "nodeName" : "mari-12.localnet.com",
    "serviceName" : "SMB",
    "state" : "running",
    "healthState" : "HEALTHY"
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Example 2:

The following example lists the details of the NFS service that is hosted on a specific node:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/nodes/mari-13.localnet.com/services/nfs'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

The **serviceStatus** array returns the details of the NFS service on the specified node.

```
{
  "serviceStatus" : [ {
    "nodeName" : "mari-13.localnet.com",
    "serviceName" : "NFS",
    "state" : "running",
    "healthState" : "HEALTHY"
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

Related reference

“mmces command” on page 133
Manage CES (Cluster Export Services) configuration.

Nodes/{name}/services/{serviceName}: PUT

Starts or stops a service on a node or node class.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `nodes/name/services/serviceName` request starts or stops a service that is hosted on a node or node class. For more information about the fields in the returned data structure, see "mmces command" on page 133.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nodes/name/services/
serviceName
```

where

nodes/name
Specifies the node or node class on which the service is hosted.

/services/serviceName
Specifies the target service of the request.

Request headers

```
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 159. List of parameters

Parameter name	Description and applicable keywords	Required/optional
name	Name of the node or node class on which the service is hosted.	Required.
serviceName	Name of the service about which the information is required.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
  "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
```

```
   "performanceData": {}
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"action":
The action to be performed on the service. You can either start or stop the service.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
   ],
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":"*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example stops the NFS service that is hosted on the node *scale01*.

Request data:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/nodes/scale01/services/nfs'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000001,
    "status" : "RUNNING",
    "submitted" : "2018-03-20 12:54:01,922",
    "completed" : "N/A",
    "runtime" : 3,
    "request" : {
      "type" : "PUT",
      "url" : "/scalemgmt/v2/nodes/scale01/services/nfs"
    },
    "result" : { },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing."
  }
}
```

Related reference

“mmces command” on page 133
Manage CES (Cluster Export Services) configuration.

NSDs: GET

Gets information about NSDs that are part of the system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The GET nsds request gets information about NSDs that are configured in the system. For more information about the fields in the data structures that are returned, see the topics “mmcrnsd command” on page 335 , “mmlsdisk command” on page 497, and “mmlsnsd command” on page 522.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nsds
```

where

nsds
Specifies NSDs as the resource of the GET call. Required.

Request headers

```
Accept: application/json
```

Request data

No request data.

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 160. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
     },
   "paging":
    {
    "next": "URL"
    },nsds":
```

```
     [
      {
         "name": "Name"
         "fileSystem": FilesystemName
         "failureGroup": "Failure Group",
         "type": "Type",
         "storagePool": "Storage Pool"
         "status": "Status"
         "availability": Availability
         "quorumDisk": "Quorum Disk",
         "size": "Size",
         "availableBlocks": "Available Blocks"
         "availableFragments": "Available Fragments"
         "nsdServers": "NSD Servers"
         "nsdVolumeId": NSD Volume ID
         "oid": "OID",
       }],
 }
```

The details of these parameters are given in the following list:

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"nsds":
An array of information about NSDs. The array contains elements that describe the NSDs. For more information about the fields in this structure, see the links at the end of this topic.

"name": "*Name*"
The name of the disk.

"fileSystem": *FilesystemName*
The file system to which the NSD belongs.

"failureGroup": "*Failure Group*"
ID of the failure group to which the NSD belongs.

"type": "*Type*"
The type of data to be stored on the disk.

"storagePool": "*Storage Pool*"
The pool to which the disk belongs.

"status": "*Status*"
Status of the disk.

"quorumDisk": "*Quorum Disk*"
The quorum status of the disk.

"size": "*Size*"
Size of the disk.

"availableBlocks": "*Available Blocks*"
Available blocks in the disk.

"availableFragments": "*Available Fragments*"
Available fragments of the disk.

"nsdServers": "*NSD Servers*"
The server that manages the I/O.

"nsdVolumeId": *NSD Volume ID*
ID of the NSD volume.

"oid": "*OID*"
Internal identifier that is used for paging.

Examples

The following example gets information about the NSDs that are configured in the system.

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/nsds'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "nsds" : [ {
      "fileSystem" : "gpfs0",
      "name" : "disk1"
     }, {
       "fileSystem" : "gpfs0",
      "name" : "disk8"
  }, {
      "fileSystem" : "objfs",
      "name" : "disk2"
  }, {
      "fileSystem" : "gpfs1",
      "name" : "disk3"
  }, {
      "fileSystem" : "gpfs1",
      "name" : "disk4"
  }, {
      "fileSystem" : "gpfs1",
      "name" : "disk5"
  }, {
      "name" : "disk6"
  }, {
      "fileSystem" : "objfs",
      "name" : "disk7"
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully"
  }
}
```

Using the field parameter ":all:" returns entire details of the NSDs that are available in the cluster. For example:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/nsds?fields=:all:'
```

```
{
  "nsds" : [ {
      "availability" : "up",
      "availableBlocks" : "9.15 GB",
      "availableFragments" : "552 kB",
      "failureGroup" : "1",
      "fileSystem" : "gpfs0",
      "name" : "disk1",
      "nsdServers" : "mari-11.localnet.com,mari-15.localnet.com,mari-14.localnet.com,
                      mari-13.localnet.com,mari-12.localnet.com",
      "nsdVolumeId" : "0A00640B58B82A8C",
      "quorumDisk" : "no",
      "remarks" : "desc",
      "size" : " 10.00GiB",
      "status" : "ready",
      "storagePool" : "system",
      "type" : "nsd"
```

```
 }, {
    "availability" : "up",
    "availableBlocks" : "9.45 GB",
    "availableFragments" : "664 kB",
    "failureGroup" : "1",
    "fileSystem" : "gpfs0",
    "name" : "disk8",
    "nsdServers" : "mari-11.localnet.com,mari-12.localnet.com,mari-13.localnet.com,
                    mari-14.localnet.com,mari-15.localnet.com",
    "nsdVolumeId" : "0A00640B58B82AD9",
    "quorumDisk" : "no",
    "remarks" : "desc",
    "size" : " 10.00GiB",
    "status" : "ready",
    "storagePool" : "data",
    "type" : "nsd"
}, {
    "availability" : "up",
    "availableBlocks" : "9.30 GB",
    "availableFragments" : "504 kB",
    "failureGroup" : "1",
    "fileSystem" : "objfs",
    "name" : "disk2",
    "nsdServers" : "mari-11.localnet.com,mari-12.localnet.com,mari-13.localnet.com,
                    mari-14.localnet.com,mari-15.localnet.com",
    "nsdVolumeId" : "0A00640B58B82A96",
    "quorumDisk" : "no",
    "remarks" : "desc",
    "size" : " 10.00GiB",
    "status" : "ready",
    "storagePool" : "system",
    "type" : "nsd"
}, {
    "availability" : "up",
    "availableBlocks" : "9.42 GB",
    "availableFragments" : "712 kB",
    "failureGroup" : "1",
    "fileSystem" : "gpfs1",
    "name" : "disk3",
    "nsdServers" : "mari-11.localnet.com,mari-15.localnet.com,mari-14.localnet.com,
                    mari-13.localnet.com,mari-12.localnet.com",
    "nsdVolumeId" : "0A00640B58B82AA1",
    "quorumDisk" : "no",
    "remarks" : "desc",
    "size" : " 10.00GiB",
    "status" : "ready",
    "storagePool" : "system",
    "type" : "nsd"
}, {
    "availability" : "up",
    "availableBlocks" : "9.94 GB",
    "availableFragments" : "504 kB",
    "failureGroup" : "1",
    "fileSystem" : "gpfs1",
    "name" : "disk4",
    "nsdServers" : "mari-11.localnet.com,mari-12.localnet.com,mari-13.localnet.com,
                   mari-14.localnet.com,mari-15.localnet.com",
    "nsdVolumeId" : "0A00640B58B82AAC",
    "quorumDisk" : "no",
    "remarks" : "desc",
    "size" : " 10.00GiB",
    "status" : "ready",
    "storagePool" : "data",
    "type" : "nsd"
}, {
    "availability" : "up",
    "availableBlocks" : "9.94 GB",
    "availableFragments" : "536 kB",
    "failureGroup" : "1",
    "fileSystem" : "gpfs1",
    "name" : "disk5",
    "nsdServers" : "mari-11.localnet.com,mari-12.localnet.com,mari-13.localnet.com,
                    mari-14.localnet.com,mari-15.localnet.com",
    "nsdVolumeId" : "0A00640B58B82AB7",
    "quorumDisk" : "no",
    "remarks" : "desc",
    "size" : " 10.00GiB",
    "status" : "ready",
    "storagePool" : "data",
    "type" : "nsd"
}, {
    "availability" : "unknown",
```

```
      "availableBlocks" : "Not Available",
      "availableFragments" : "Not Available",
      "failureGroup" : "1",
      "name" : "disk6",
      "nsdServers" : "mari-11.localnet.com,mari-15.localnet.com,mari-12.localnet.com,
                      mari-13.localnet.com,mari-14.localnet.com",
      "nsdVolumeId" : "0A00640B58B82AC2",
      "quorumDisk" : "no",
      "size" : " 10.00GiB",
      "status" : "ready",
      "storagePool" : "data",
      "type" : "nsd"
  }, {
       "availability" : "up",
      "availableBlocks" : "7.22 GB",
      "availableFragments" : "4.62 MB",
      "failureGroup" : "1",
      "fileSystem" : "objfs",
      "name" : "disk7",
      "nsdServers" : "mari-11.localnet.com,mari-12.localnet.com,mari-13.localnet.com,
                      mari-14.localnet.com,mari-15.localnet.com",
      "nsdVolumeId" : "0A00640B58B82ACD",
      "quorumDisk" : "no",
      "remarks" : "desc",
      "size" : " 10.00GiB",
      "status" : "ready",
      "storagePool" : "data",
      "type" : "nsd"
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully"
  }
}
```

Related reference

“mmcrnsd command” on page 335
Creates Network Shared Disks (NSDs) used by GPFS.

“mmlsnsd command” on page 522
Displays Network Shared Disk (NSD) information for the GPFS cluster.

“mmlsdisk command” on page 497
Displays the current configuration and state of the disks in a file system.

NSDs/{nsdName}: GET

Gets information about a specific NSD that is part of the system.

Availability

Available on all IBM Spectrum Scale editions.

Description

The GET nsds/nsdName request gets information about an NSD that is configured in the system. For more information about the fields in the data structures that are returned, see the topics “mmcrnsd command” on page 335 , “mmlsdisk command” on page 497, and “mmlsnsd command” on page 522.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/nsds/nsdName
```

where

nsds/nsdName
Specifies the NSD about which you need to get the details. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 161. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
nsdName	Name of the NSD.	Required.

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
     },
   "paging":
    {
    "next": "URL"
    },nsds": [
    {
         "name": "Name"
         "fileSystem": FilesystemName
         "failureGroup": "Failure Group",
         "type": "Type",
```

```
          "storagePool": "Storage Pool"
          "status": "Status"
          "availability": Availability
          "quorumDisk": "Quorum Disk",
          "size": "Size",
          "availableBlocks": "Available Blocks"
          "availableFragments": "Available Fragments"
          "nsdServers": "NSD Servers"
          "nsdVolumeId": NSD Volume ID
          "oid": "OID",
        }],
  }
```

The details of these parameters are given in the following list:

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"nsds":
An array of information about NSDs. The array contains elements that describe the NSDs. For more information about the fields in this structure, see the links at the end of this topic.

"name": "*Name*"
The name of the disk.

"fileSystem": *FilesystemName*
The file system to which the NSD belongs.

"failureGroup": "*Failure Group*"
ID of the failure group to which the NSD belongs.

"type": "*Type*"
The type of data to be stored on the disk.

"storagePool": "*Storage Pool*"
The pool to which the disk belongs.

"status": "*Status*"
Status of the disk.

"quorumDisk": "*Quorum Disk*"
The quorum status of the disk.

"size": "*Size*"
Size of the disk.

"availableBlocks": "*Available Blocks*"
Available blocks in the disk.

"availableFragments": "*Available Fragments*"
Available fragments of the disk.

"nsdServers": "*NSD Servers*"
The server that manages the I/O.

"nsdVolumeId": *NSD Volume ID*
ID of the NSD volume.

"oid": "*OID*"
Internal identifier that is used for paging.

Examples

The following example gets information about the NSD nsd1 that are configured in the system.

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/nsds/nsd1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/nsds/nsd1?lastId=1001"
  },
  "nsds": [
    {
        "name": "nsd1",
        "fileSystem": "gpfs0",
        "failureGroup": "1",
        "type": "dataOnly",
        "storagePool": "data",
        "status": "ready",
        "availability": "up",
        "quorumDisk": "no",
        "remarks": "This is a comment",
        "size": "10.00 GB",
        "availableBlocks": "730.50 MB",
        "availableFragments": "1.50 MB",
        "nsdServers": "gpfsgui-21.localnet.com",
        "nsdVolumeId": "0A0064155874F5AA"
    }
  ]
}
```

Related reference

“mmcrnsd command” on page 335
Creates Network Shared Disks (NSDs) used by GPFS.

“mmlsnsd command” on page 522
Displays Network Shared Disk (NSD) information for the GPFS cluster.

“mmlsdisk command” on page 497
Displays the current configuration and state of the disks in a file system.

Perfmon/data: GET

Gets the performance data from the cluster.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET perfmon/data` request gets performance details from the cluster with the help of queries. The query is written in the performance monitoring tool query language format. For example, *query copu_user metric for last 10 seconds*. For more information about the fields in the data structures that are returned, see “Querying performance data by using /perfmon/data request ” on page 1556.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/perfmon/data
```

where

perfmon/data
Specifies the performance monitoring tool as the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 162. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
query	Query to fetch the performance details. The query is written in the performance monitoring tool query language format.	Required.

Request data

No request data.

Response data

```
{
  "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "performanceData": {}
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"performanceData":
The performance data that is returned from the performance monitoring tool based on the query that you entered.

Examples

The following example shows how to use the API command to get the performance details when you use the following query: *metrics avg(cpu_user) last 30 bucket_size 1*

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/perfmon/data?query=metrics%20avg(cpu_user)
%20last%2030%20bucket_size%201'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": 200,
    "message": "..."
  },
  "performanceData": "null"
}
```

"mmperfmon command" on page 590
Configures the Performance Monitoring tool and lists the performance metrics.

"Querying performance data by using /perfmon/data request " on page 1556
The GET `perfmon/data` request gets performance details from the cluster with the help of queries. A query is written in the performance monitoring tool query language format.

Querying performance data by using /perfmon/data request

The GET `perfmon/data` request gets performance details from the cluster with the help of queries. A query is written in the performance monitoring tool query language format.

The types of queries that can be used to get the performance details are as follows:

- "Query by metric specification" on page 1556
- "Query by key specification" on page 1557
- "Query by group specification" on page 1557

Query by metric specification

A metric specification is a list of metrics and operation on metrics that are separated by commas. For example, metrics *cpu_idle,cpu_system* and *sum(cpu_user)*

Synopsis: *metrics metric1,... [key_filter_specs] time_filter [grouping_spec] bucket_spec*

Metric names are based on the types of sensors that are available in the performance monitoring tool. For more information on the available performance metrics, see *List of performance metrics* topic in the *IBM*

Spectrum Scale: Concepts, Planning, and Installation Guide. A query can contain metrics from different sensors. You can perform the following operations on the performance data that is retrieved by the query:

- sum
- avg
- max
- min
- rate

For example, *sum (netdev_bytes_r)* returns the sum of all *netdev_bytes_r* values covered by the query. The rate operation returns a rate of the data that is retrieved by the query. For example, *rate (netdev_bytes_r)* returns bytes received per second. You can vary the size of the bucket.

Query by key specification

You can use various keys that identify the requested metrics. A key is a set of metric names that are separated by the "|" character. Keys are based on available sensor instances in the performance monitoring tool setup. For more information on how to find out the key names by using the mmperfmon query, see "mmperfmon command" on page 590.

Synopsis: *key key1,... [key_filter_specs] time_filter [grouping_spec] bucket_spec*

Example: *key device1|CPU|cpu_idle, device2|Network|eth0|netdev_bytes_r*

Regular expressions can be used in a key to increase the range of data returned. For example, *key device[1-4]|CPU|cpu_idle*.

Query by group specification

The group query retrieves metric data by using buckets of given time length, for a given time period, and filtered keys. Optionally, grouping can be done according to the similarities in the keys. For example, you can specify a sensor name to return all metrics for that group of sensors.

Synopsis: *group sensor [key_filter_specs] time_filter [grouping_spec] bucket_spec*

The following arguments are mandatory in a query by group:

- *get_list*: Specification of a list of requested metrics, keys, or a sensor.
- *time_filter*: Time span of the request.
- *bucket_spec*: Time granularity of the returned data.

Using filters and grouping in a query

You can use various filters and grouping techniques to get the exact details that you are looking for.

Key filter specification

Each metric can contain a key filter specification that can be used to narrow down the results that are returned for the metrics specified in the query. If the query is specified by a key specification, the key filter specification is ignored.

The key filter specification consists of the key word "from", followed by a sequence of assignments of the form *metric_name = string_value*. These are separated by commas. The *metric_name* must be an identifier of metric type *SEMTYPE_IDENTIFIER*. The commas imply a *logical AND* operation of the assignments and these assignments apply to all keys where the *metric_name* is used. The *string_value* can be a regular expression, permitting additional filtering of values returned by the query. Such a key filter specification is given as follows:

... from netdev_name=eth[01] ...

For example, the metric *cpu_user* has the following key structure: *node|sensor|metric_name*. In this case, an instance of *cpu_user* can have the key *server1|CPU|cpu_user*. If the *key_filter_specs* is omitted, then the values for the metric *cpu_user* is searched by using the key containing wildcards. For example,

||*cpu_user*. This results in returning values from all reporting CPU sensors. If the *key_filter_specs* is present, for example, *from node=server1*, then only the value from the sensor on server1 is returned (key is *server1|*|cpu_user*).

Time filter specification

Each query must contain a time period for which data is to be returned. The time period for the query can be specified as a time span such as last n buckets, last t seconds, and the current value. The last 'n' filter returns the last 'n' buckets of metric data. The duration 't' filter returns a list of buckets of metric data that covers last 't' seconds of time. The 'now' filter returns the current value (last bucket) for given metrics.

Only one of these types can be used at a time.

Time span consists of a starting time (tstart) and an end time (tend), both expressed either as the "unix time" in seconds or a combination of date and time specification. For example, "2012-11-10 13:00:00".

If only *tstart* is specified, then the query covers all data from that time until now. If only *tend* is specified, a default number of buckets up to the specified time is returned. The default number of buckets is 12.

The time values specified as part of the filter specification can be rounded to fit the bucket sizes that are used internally to store the metrics.

Grouping specification

A grouping is used to split results of metric operations based on specified key metric names. It consists of the keyword *group_by* followed by a comma-separated list of metric names. Grouping is done based on the key value that is associated with the metric name specified in the grouping specification. A metric name in the *key_filter_spec* must be of type *SEMTYPE_IDENTIFIER*.

An example for grouping: If the metric operation is *sum(netdev_bytes_r)* and the *grouping_key* is *netdev_name*. Then, the result will be a list of sums of values of *netdev_bytes_r*. For example, eth0, eth1, and lo0 – 3 sums.

Bucket specification

A query must contain a bucket specification. It indicates the time interval in seconds to which the data is accumulated to. For example, specifying *bucket_size* 30 returns the data accumulated in 30-second intervals. Depending on the type of metric, metrics are accumulated in different ways. For example, averaged for averages and summed up for counters.

Query samples

metrics cpu_user from node=anaphera-dev2 last 10 bucket_size 1

Gets metric *cpu_user* for the instance where the key element node equals *anaphora-dev2* and return the last 10 buckets of a 1-second size.

metrics cpu_user, cpu_system, cpu_idle from node=anaphera-dev2

tstart 2012-11-10 08:00:00 tend 2012-11-10 18:00:00 bucket_size 30

Gets metrics *cpu_user, cpu_system* and *cpu_idle* for node *anaphora-dev3* from given start to end time, using a bucket size of 30 seconds.

metrics sum(cpu_user), sum(netdev_bytes_r) last 30 bucket_size 10

Gets the sum of *cpu_user* and *netdev_bytes_r* metrics for all instances and the last 30 buckets by using a bucket size of 10 seconds.

metrics sum(netdev_bytes_r) last 50 group_by netdev_name bucket_size 1

Gets sum of *netdev_bytes_r* (last 50 buckets of size 1 second), grouped by key attribute *netdev_name*.

key anaphera-dev2|CPU|cpu_user last 10 bucket_size 1

Gets metric *cpu_user* for the instance specified by the explicit key by using bucket size equal to 1, for 10 seconds (result will contain 10 buckets)

metrics cpu_user duration 210 bucket_size 100

Gets metric *cpu_user* for all nodes and a duration that covers last 210 seconds of time using a bucket size 100. Therefore, the result consists of three buckets.

group CPU last 30 bucket_size 1

Gets all metrics of CPU sensor for all nodes by using bucket size 1 for last 10 buckets (equal to last 10 seconds).

Perfmon/sensors/{sensorName}: GET

Gets the configuration details of a specific performance monitoring sensor.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET perfmon/sensors/{sensorName}` request gets details of a specific performance monitoring sensor that is configured in the cluster. For more information about the fields in the data structures that are returned, see the topic "mmperfmon command" on page 590.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/perfmon/sensors/{sensorName}
```

where

perfmon/sensors/{sensorName}
Specifies the a particular performance monitoring sensor as the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 163. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
sensorName	Name of the performance monitoring sensor.	Required.

Request data

No request data.

Response data

```
{
  "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "performanceData": {}
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"performanceData":
The configuration details of the performance monitoring sensor.

Examples

The following example shows how to use the API command to get the details of the performance monitoring sensor *Netstat*.

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/perfmon/sensors/TCTDebugLweDestroyStats'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "sensorConfig" : [ {
    "component" : "TCT",
    "defaultPeriod" : 10,
    "defaultRestriction" : "",
    "description" : "",
    "enabledPerDefault" : true,
    "generic" : true,
    "minimumPeriod" : 1,
    "period" : 10,
    "restrict" : [ ],
    "restrictionType" : "USERNODECLASS",
    "sensorName" : "TCTDebugLweDestroyStats"
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

“mmperfmon command” on page 590
Configures the Performance Monitoring tool and lists the performance metrics.

Perfmon/sensors: GET

Gets the performance sensor configuration.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET perfmon/sensors` request gets details of the performance monitoring sensors that are configured in the cluster. For more information about the fields in the data structures that are returned, see the topic "mmperfmon command" on page 590.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/perfmon/sensors
```

where

perfmon/sensors
Specifies the performance monitoring sensors as the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 164. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Request data

No request data.

Response data

```
{
  "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "performanceData": {}
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"performanceData":
The performance data that is returned from the performance monitoring tool based on the query that you entered.

Examples

The following example shows how to use the API command to get the details of the performance monitoring sensor configuration in the cluster.

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/perfmon/sensors'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "sensorConfig" : [ {
    "component" : "TCT",
    "defaultPeriod" : 10,
    "defaultRestriction" : "",
    "description" : "",
    "enabledPerDefault" : true,
    "generic" : true,
    "minimumPeriod" : 1,
    "period" : 20,
    "restrict" : [ ],
    "restrictionType" : "USERNODECLASS",
    "sensorName" : "TCTDebugLweDestroyStats"
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully."
  }
}
```

“mmperfmon command” on page 590
Configures the Performance Monitoring tool and lists the performance metrics.

Perfmon/sensors/{sensorName}: PUT

Configures the performance monitoring sensor.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `perfmon/sensors/sensorName` request configures the sensor. For more information about the fields in the data structures that are returned, see “mmperfmon command” on page 590.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/perfmon/sensors/sensorName
```

where

perfmon/sensors/sensorName
Specifies the sensor that you need to modify. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Parameters

The following parameters can be used in the request URL to customize the request:

Table 165. List of parameters

Parameter name	Description and applicable keywords	Required/optional
sensorName	The name of the file system that needs to be configured.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
  "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "performanceData": {}
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"performanceData":
The configuration details of the performance monitoring sensor.

Response data

```
{
   "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "jobs": [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
   ],
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":"*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example sets nodes or node class where you need to run the sensor TCTDebugLweDestroyStats and it also sets the period of the sensor.

Request data:

```
{
  "period": 20
}
```

Corresponding request URL:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
-d '{ \
    "period": 20 \
 }'
'https://198.51.100.1:443/scalemgmt/v2/perfmon/sensors/TCTDebugLweDestroyStats'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "jobs" : [ {
    "jobId" : 1000000000001,
    "status" : "RUNNING",
    "submitted" : "2018-03-23 14:55:38,385",
    "completed" : "N/A",
    "runtime" : 14,
    "request" : {
      "data" : {
        "period" : 20
      },
      "type" : "PUT",
      "url" : "/scalemgmt/v2/perfmon/sensors/TCTDebugLweDestroyStats"
    },
    "result" : { },
    "pids" : [ ]
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing."
  }
}
```

“mmperfmon command” on page 590
Configures the Performance Monitoring tool and lists the performance metrics.

Remotemount/authenticationkey: GET

Gets the public RSA authentication key of a cluster that is used for remote mounting. This API can be run on the cluster that owns the file systems and the cluster that remotely mounts file systems. The public authentication key can be used as input for POST and PUT request of the endpoints **/remotemount/owningclusters** and **/remotemount/remoteclusters**.

Availability

Available on all IBM Spectrum Scale editions.

Note: Only the users with user roles *Administrator* or *Container Operator* have permission to use this REST API endpoint.

Description

The GET /remotemount/authenticationkey request gets the public RSA authentication key of a cluster that is used for remote mounting. For more information about the fields in the data structures that are returned, see "mmauth command" on page 96.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/remotemount/
authenticationkey
```

where

remotemount/authenticationkey
Specifies the target of this GET request.

Request headers

```
Accept: application/json
```

Request data

No request data.

Response data

```
{
  "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   }
  "key": "Active public RSA key",
  "newKey": "New public RSA key",
  "ciphers": "Ciphers of the RSA key",
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"key": "*Active public RSA key*"
The currently active public RSA key.

"newKey": "*New public RSA key*"
A newly generated public RSA key. This key becomes active when it is committed. To commit this new key, use the endpoint **PUT /scalemgmt/v2/remotemount/authenticationkey**.

"ciphers": *Ciphers of the RSA key*"
The list of ciphers of the RSA key. It sets the security mode for communications between the current cluster and the remote cluster. For more information on the list of ciphers, see "mmauth command" on page 96.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example gets the public RSA authentication key that is required for remote mounting.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/remotemount/authenticationkey'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": 200,
    "message": "..."
  },
  "key": [
    "string"
  ],
  "newKey": [
    "string"
  ],
  "ciphers": "[ 'AES128-SHA', 'AES256-SHA' ]"
}
```

Related reference

"mmauth command" on page 96
Manages secure access to GPFS file systems.

Remotemount/authenticationkey: POST

Generates a new private or public RSA authentication key pair on a cluster that is used for remote mounting. This API can be run on the cluster that owns the file systems and the cluster that remotely mounts file systems.

Availability

Available on all IBM Spectrum Scale editions.

Note: Only the users with user roles *Administrator* or *Container Operator* have permission to use this REST API endpoint.

Description

The POST `/remotemount/authenticationkey` request generates a new RSA authentication key pair (private and public) on a cluster that is used for remote mounting. Use **`GET /scalemgmt/v2/remotemount/authenticationkey`** to check whether a new authentication key is already generated. The new key is in addition to the currently active committed key. Both keys are accepted until the administrator runs the **`mmauth genkey commit`** command or **`PUT /scalemgmt/v2/remotemount/authenticationkey`** endpoint.

For more information about the fields in the data structures that are returned, see "mmauth command" on page 96.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/remotemount/
authenticationkey
```

where

remotemount/authenticationkey
: Specifies the target of this POST request.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

None.

Request data

None.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
```

```
            "exitCode":"Exit code",
            "stderr":"Error",
            "stdout":"String",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted":"Time",
        "completed":Time",
        "status":"Job status",
        }
      ],
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*'
Array of commands that are run in this job.

"progress":"*String*'
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to generate a new RSA authentication key pair.

Request URL:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/remotemount/authenticationkey'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": ["''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": ["''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": ["''EFSSG0740C There are not enough resources available to create
                   a new independent file set.'',  ...]",
        "stdout": ["''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "POST",
        "url": "/scalemgmt/v2/remotemount/authenticationkey",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

Related reference

“mmauth command” on page 96
Manages secure access to GPFS file systems.

Remotemount/authenticationkey: PUT

Commits or propagates a new RSA authentication key on a cluster that is used for remote mounting. This API can be run on the cluster that owns the file systems and the cluster that remotely mounts file systems.

Availability

Available on all IBM Spectrum Scale editions.

Note: Only the users with user roles *Administrator* or *Container Operator* have permission to use this REST API endpoint.

Description

The PUT `/remotemount/authenticationkey` request commits or propagates a new RSA authentication key on a cluster that is used for remote mounting. Use the POST `/scalemgmt/v2/remotemount/authenticationkey` request to generate a new RSA key pair (private and public key) for remote mounting. For more information about the fields in the data structures that are returned, see "mmauth command" on page 96.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/remotemount/
authenticationkey
```

where

remotemount/authenticationkey
Specifies the target of the PUT request.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 166. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
action	The action to perform on the authentication key. Possible values are: `commit` or `propagate`.	Required.
nodes	Comma-separated list of nodes or node classes that defines the nodes on which to commit or propagate the authentication key.	Required.

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*'
Array of commands that are run in this job.

"progress":"*String*'
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following API endpoint shows how to commit authentication key on the node CESnode1.

Request URL:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/remotemount/authenticationkey?
action=commit&nodes=CESnode1'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": "[''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": "[''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": "[''EFSSG0740C There are not enough resources available to create
                  a new independent file set.'',  ...]",
        "stdout": "[''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "POST",
        "url": "/scalemgmt/v2/remotemount/authenticationkey?action=commit&nodes=CESnode1",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

Related reference

“mmauth command” on page 96
Manages secure access to GPFS file systems.

Remotemount/owningclusters: GET

Lists the clusters that own file systems that can be mounted remotely. This API must be run on a cluster that mounts file systems remotely.

Availability

Available on all IBM Spectrum Scale editions.

Description

The GET `/remotemount/owningcluster` lists the clusters that own file systems that can be mounted remotely. For more information about the fields in the data structures that are returned, see “mmremotecluster command” on page 660.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/remotemount/owningclusters
```

where

remotemount/owningclusters
Specifies target of the GET request.

Request headers

```
Accept: application/json
```

Request data

No request data.

Response data

```
{
  "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   }
   "owningClusters":
    {
      "owningCluster": "Name of the cluster",
      "contactNodes": "Contact nodes of owning cluster",
      "keyDigest": "SHA digest of the public key",
      "filesystemPairs": "File system mapping",
    }
   "filesystemPair"
     {
      "owningClusterFilesystem ": "File system on the owning cluster",
      "owningClusterFilesystem ": "File system on the remote cluster",
     }
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"owningClusters":
Owning cluster details.

"owningCluster": "*Name of the cluster*"
The owning cluster of the remote file system.

"contactNodes": "*Contact nodes of owning cluster*"
The contact nodes of the owning cluster used for remote mounting.

"keyDigest": "*SHA digest of the public key*"
The SHA digest of the public key of the owning cluster.

"filesystemPairs": "*File system mapping*"
The mapping of file systems of owning cluster and remote cluster.

"filesystemPair":
File system pair details.

"owningClusterFilesystem ": "*File system on the owning cluster*"
The file system on the owning cluster.

"owningClusterFilesystem ": "*File system on the remote cluster*"
The file system on the remote cluster.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example shows how to get the list of the clusters that own file systems that can be mounted remotely.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/remotemount/owningclusters'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": 200,
    "message": "..."
  },
   "owningClusters": [
    {
      "owningCluster": "string",
      "contactNodes": [
        "string"
      ],
      "keyDigest": "string",
      "filesystemPairs": "{ {'owningClusterFilesystem':'fs1',
'remoteClusterFilesystem':'fs1_remote'}, {'owningClusterFilesystem':'fs2',
'remoteClusterFilesystem':'fs2_remote'} }"
    }
  ]
}
```

Related reference

“mmremotecluster command” on page 660
Manages information about remote GPFS clusters.

Remotemount/owningclusters: POST

Registers a cluster that owns file systems that can be mounted remotely. This API must be run on a cluster that mounts file systems remotely.

Availability

Available on all IBM Spectrum Scale editions.

Note: Only the users with user roles *Administrator* or *Container Operator* have permission to use this REST API endpoint.

Description

The POST `remotemount/owningclusters` request registers a cluster that own file systems that can be mounted remotely. For more information about the fields in the data structures that are returned, see "mmremotecluster command" on page 660.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/remotemount/owningclusters
```

where

remotemount/owningclusters
Specifies the target of the operation. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 167. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
  "owningCluster": "Name of the cluster",
  "contactNodes": "List of contact nodes",
  "key": "Public RSA key"
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"owningCluster":"*Name of the cluster*"
The owning cluster of the remote file system.

"contactNodes":"*List of contact nodes*"
The contact nodes of the owning cluster that is used for remote mounting.

"key":"*Public RSA key*"
The public RSA key of the owning cluster.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*'
Array of commands that are run in this job.

"progress":"*String*'
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to register a cluster that owns file systems that can be mounted remotely.

Request URL:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
-d '{ \
   "owningCluster": "string", \
   "contactNodes": "[%27node1%27, %27node2%27]", \
   "key": [ \
     "string" \
   ] \
 }'
'https://198.51.100.1:443/scalemgmt/v2/remotemount/owningclusters'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": "[''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": "[''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": "[''EFSSG0740C There are not enough resources available to create
                  a new independent file set.'',  ...]",
        "stdout": "[''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
```

```
      "request": {
        "type": "POST",
        "url": "/scalemgmt/v2/remotemount/owningclusters",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

“mmremotecluster command” on page 660
Manages information about remote GPFS clusters.

Remotemount/owningclusters/{owningCluster}: DELETE

Removes the registration of a cluster that owns file systems that can be mounted remotely. This API must be run on a cluster that mounts file systems remotely.

Availability

Available on all IBM Spectrum Scale editions.

Note: Only the users with user roles *Administrator* or *Container Operator* have permission to use this REST API endpoint.

Description

The `DELETE remotemount/owningclusters/{owningCluster}` request removes the registration of a cluster that own file systems that can be mounted remotely. For more information about the fields in the data structures that are returned, see "mmremotecluster command" on page 660.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/remotemount/owningclusters/
{owningCluster}
```

where

remotemount/owningclusters/*{owningCluster}*
: Specifies the cluster that needs to be unregistered. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 168. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
owningCluster	Name of the cluster that must be unregistered.	Required.

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
```

```
            "progress":"String,
            "exitCode":"Exit code",
            "stderr":"Error",
            "stdout":"String",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted":"Time",
        "completed":Time",
        "status":"Job status",
        }
      ],
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to remove the registration of the cluster *Cluster1*.

Request URL:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/remotemount/owningclusters/Cluster1'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": "[''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": "[''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": "[''EFSSG0740C There are not enough resources available to create
                  a new independent file set.'',  ...]",
        "stdout": "[''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "DELETE",
        "url": "/scalemgmt/v2/remotemount/owningclusters/Cluster1",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

“mmremotecluster command” on page 660
Manages information about remote GPFS clusters.

Remotemount/owningclusters/{owningCluster}: GET

Gets the details of the cluster that owns file systems that can be mounted remotely. This API must be run on a cluster that mounts file systems remotely.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET /remotemount/owningclusters/{owningCluster}` gets the details of the cluster that owns the file systems that can be mounted remotely. For more information about the fields in the data structures that are returned, see “mmremotecluster command” on page 660.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/remotemount/owningclusters/
{owningCluster}
```

where

remotemount/owningclusters/{owningCluster}
Specifies the target of the GET request.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 169. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
owningCluster	Name of the owning cluster.	Required.

Request data

No request data.

Response data

```
{
  "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   }
   "owningClusters":
    {
      "owningCluster": "Name of the cluster",
      "contactNodes": "Contact nodes of owning cluster",
      "keyDigest": "SHA digest of the public key",
      "filesystemPairs": "File system mapping",
    }
   "filesystemPair"
     {
      "owningClusterFilesystem ": "File system on the owning cluster",
      "owningClusterFilesystem ": "File system on the remote cluster",
```

```
    }
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"owningClusters":
Owning cluster details.

"owningCluster": "*Name of the cluster*"
The owning cluster of the remote file system.

"contactNodes": "*Contact nodes of owning cluster*"
The contact nodes of the owning cluster used for remote mounting.

"keyDigest": "*SHA digest of the public key*"
The SHA digest of the public key of the owning cluster.

"filesystemPairs": "*File system mapping*"
The mapping of file systems of owning cluster and remote cluster.

"filesystemPair":
File system pair details.

"owningClusterFilesystem": "*File system on the owning cluster*"
The file system on the owning cluster.

"owningClusterFilesystem": "*File system on the remote cluster*"
The file system on the remote cluster.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example shows how to get the details of the cluster *Cluster1* that owns file systems that can be mounted remotely.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/remotemount/owningclusters/Cluster1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": 200,
    "message": "..."
  },
   "owningClusters": [
    {
      "owningCluster": "Cluster1",
      "contactNodes": [
        "string"
      ],
      "keyDigest": "string",
      "filesystemPairs": "{ {'owningClusterFilesystem':'fs1',
'remoteClusterFilesystem':'fs1_remote'}, {'owningClusterFilesystem':'fs2',
'remoteClusterFilesystem':'fs2_remote'} }"
    }
```

```
  ]
}
```

Related reference

“mmremotecluster command” on page 660
Manages information about remote GPFS clusters.

Remotemount/owningclusters/{owningCluster}: PUT

Update registration of a cluster that owns file systems that can be mounted remotely. This API must be run on a cluster that mounts file systems remotely.

Availability

Available on all IBM Spectrum Scale editions.

Note: Only the users with user roles *Administrator* or *Container Operator* have permission to use this REST API endpoint.

Description

The PUT `remotemount/owningclusters/{owningCluster}` request update registration of a cluster that owns file systems that can be mounted remotely. For more information about the fields in the data structures that are returned, see "mmremotecluster command" on page 660.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/remotemount/owningclusters/
{owningCluster}
```

where

remotemount/owningclusters/*{owningCluster}*
: Specifies the cluster for which the registration needs to be updated. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 170. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
owningCluster	Name of the target cluster.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
"owningClusters":
{
"owningCluster": "Name of the cluster",
"contactNodes": "Contact nodes of owning cluster",
"keyDigest": "SHA digest of the public key",
"filesystemPairs": "File system mapping", }
"filesystemPair" { "owningClusterFilesystem ":
"File system on the owning cluster",
"owningClusterFilesystem ": "File system on the remote cluster"
}
```

"owningClusters":
Owning cluster details.

"owningCluster": "*Name of the cluster*"
The owning cluster of the remote file system.

"contactNodes": "*Contact nodes of owning cluster*"
The contact nodes of the owning cluster used for remote mounting.

"keyDigest": "*SHA digest of the public key*"
The SHA digest of the public key of the owning cluster.

"filesystemPairs": "*File system mapping*"
The mapping of file systems of owning cluster and remote cluster.

"filesystemPair":
File system pair details.

"owningClusterFilesystem": "*File system on the owning cluster*"
The file system on the owning cluster.

"owningClusterFilesystem": "*File system on the remote cluster*"
The file system on the remote cluster.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to update the registration of the cluster *Cluster1*.

Request URL:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
-d '{ \
   "owningCluster": "Cluster1", \
   "contactNodes": "[%27node1%27, %27node2%27]", \
   "key": [ \
     "string"
'https://198.51.100.1:443/scalemgmt/v2/remotemount/owningclusters/Cluster1'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": "[''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": "[''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": "[''EFSSG0740C There are not enough resources available to create
                  a new independent file set.'',  ...]",
        "stdout": "[''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "PUT",
        "url": "/scalemgmt/v2/remotemount/owningclusters/Cluster1",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

“mmremotecluster command” on page 660
Manages information about remote GPFS clusters.

Remotemount/remoteclusters: GET

Lists the clusters that mount file systems of an owning cluster remotely. This API must be run on a cluster that mounts file systems remotely.

Availability

Available on all IBM Spectrum Scale editions.

Description

The GET `/remotemount/remoteclusters` lists the clusters that mount file systems of an owning cluster remotely. For more information about the fields in the data structures that are returned, see “mmremotecluster command” on page 660.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/remotemount/remoteclusters
```

where

remotemount/remoteclusters
Specifies the target of this GET request.

Request headers

```
Accept: application/json
```

Request data

No request data.

Response data

```
{
  "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   }
   "remoteClusters":
    {
      "remoteCluster": "Name of the cluster",
      "ciphers": "Ciphers of the RSA key",
      "keyDigest": "SHA digest of the public key",
      "owningClusterFilesystems": "Name of the file system of the owning cluster.",
    }
   "OwningFilesystem"
     {
      "filesystem ": "File system on the owning cluster",
      "access ": "Access permissions",
     }
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"remoteClusters":
Owning cluster details.

"remoteCluster": "*Name of the cluster*"
The cluster that remotely mounts the file systems of the owning cluster

"ciphers": "*Ciphers of the RSA key*"
The list of ciphers of the RSA key. It sets the security mode for communications between the current cluster and the remote cluster. For more information on the list of ciphers, see "mmauth command" on page 96.

"keyDigest": "*SHA digest of the public key*"
The SHA digest of the public key of the remote cluster.

"owningClusterFilesystems": "*Name of the owning cluster file systems.*"
The names of file system of owning cluster and the access permissions to them. Optional.

"OwningFilesystem ":
Owning file system details.

"filesystem ": "*File system on the owning cluster*"
The name of the owning cluster filesystem You can also use the keyword 'all' when addressing all file systems of the owning cluster.

"access": "*Access permissions*"
The access permissions to the owning cluster file system.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example shows how to get the list of the clusters that mounts the file system remotely.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/remotemount/remoteclusters'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": 200,
    "message": "..."
  },
  "remoteClusters": [
    {
      "remoteCluster": "Cluster1",
      "ciphers": "[ 'AES128-SHA', 'AES256-SHA' ]",
      "keyDigest": "string",
      "owningClusterFilesystems": "{ {'name':'gpfs0', 'access':'rw'}, {'name':'gpfs1',
'access':'ro'} }"
    }
  ]
}
```

Related reference

"mmremotecluster command" on page 660
Manages information about remote GPFS clusters.

Remotemount/remoteclusters: POST

Registers a cluster that can mount one or more file systems of an owning cluster. This API must be executed on a cluster that owns file systems that must be mounted remotely.

Availability

Available on all IBM Spectrum Scale editions.

Note: Only the users with user roles *Administrator* or *Container Operator* have permission to use this REST API endpoint.

Description

The POST `remotemount/remoteclusters` request registers a cluster that can mount one or more file systems of an owning cluster. For more information about the fields in the data structures that are returned, see "mmremotecluster command" on page 660.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/remotemount/remoteclusters/
{remoteCluster}
```

where

remotemount/remoteclusters
Specifies the cluster that needs to be registered. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 171. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
"remoteClusters":
    {
      "remoteCluster": "Name of the cluster",
      "ciphers": "Ciphers of the RSA key",
      "key": "SHA digest of the public key",
      }
```

"remoteClusters":
Owning cluster details.

"remoteCluster": "*Name of the cluster*"
The cluster that remotely mounts the file systems of the owning cluster.

"ciphers": "*Ciphers of the RSA key*"
The list of ciphers of the RSA key. It sets the security mode for communications between the current cluster and the remote cluster. For more information on the list of ciphers, see "mmauth command" on page 96.

"key": "*SHA digest of the public key*"
The SHA digest of the public key of the remote cluster.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to register a cluster that can mount one or more file systems of an owning cluster.

Request URL:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
-d '{ \
   "remoteCluster": "Cluster1", \
   "contactNodes": "[%27node1%27, %27node2%27]", \
   "key": ["string"]'}'
'https://198.51.100.1:443/scalemgmt/v2/remotemount/remoteclusters'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": ["'mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": ["'(2/3) Linking fileset'']",
```

```
        "exitCode": "0",
        "stderr": ["'EFSSG0740C There are not enough resources available to create
                   a new independent file set.'',  ...]",
        "stdout": ["'EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "POST",
        "url": "/scalemgmt/v2/remotemount/remoteclusters",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

“mmremotecluster command” on page 660
Manages information about remote GPFS clusters.

Remotemount/remoteclusters/{remoteCluster}: DELETE

Removes the registration of a cluster that can mount file systems remotely. This API must be run on a cluster that owns the file systems that are no longer be mounted remotely.

Availability

Available on all IBM Spectrum Scale editions.

Note: Only the users with user roles *Administrator* or *Container Operator* have permission to use this REST API endpoint.

Description

The `DELETE remotemount/remoteclusters/{remoteCluster}` request removes the registration of a cluster that can mount one or more file systems of an owning cluster. For more information about the fields in the data structures that are returned, see "mmremotecluster command" on page 660.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/remotemount/remoteclusters/
{remoteCluster}
```

where

remotemount/remoteclusters/*{remoteCluster}*
: Specifies the cluster for which the registration needs to be removed. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 172. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
remoteCluster	Name of the target cluster.	Required.

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
```

```
            "stderr":"Error",
            "stdout":"String",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted":"Time",
        "completed":Time",
        "status":"Job status",
        }
      ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to remove the registration of the cluster *Cluster1*.

Request URL:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/remotemount/remoteclusters/Cluster1'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": "[''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": "[''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": "[''EFSSG0740C There are not enough resources available to create
                  a new independent file set.'',  ...]",
        "stdout": "[''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "DELETE",
        "url": "/scalemgmt/v2/remotemount/remoteclusters/Cluster1",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

“mmremotecluster command” on page 660
Manages information about remote GPFS clusters.

Remotemount/remoteclusters/{remoteCluster}: GET

Gets the details of the cluster that mounts file systems of an owning cluster remotely. This API must be run on a cluster that mounts file systems remotely.

Availability

Available on all IBM Spectrum Scale editions.

Description

The GET `/remotemount/remoteclusters/`*{remoteCluster}* gets the details of the cluster that mounts file systems of an owning cluster remotely. For more information about the fields in the data structures that are returned, see "mmremotecluster command" on page 660.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/remotemount/remoteclusters/
{remoteCluster}
```

where

remotemount/remoteclusters/{remoteCluster}
Specifies the target of the GET request.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 173. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
remoteCluster	Name of the remote cluster.	Required.

Request data

No request data.

Response data

```
{
  "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   }
   "remoteClusters":
    {
      "remoteCluster": "Name of the cluster",
      "ciphers": "Ciphers of the RSA key",
      "keyDigest": "SHA digest of the public key",
      "owningClusterFilesystems": "Name of the file system of the owning cluster.",
    }
   "OwningFilesystem"
     {
      "filesystem ": "File system on the owning cluster",
      "access ": "Access permissions",
```

```
    }
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"remoteClusters":
Owning cluster details.

"remoteCluster": "*Name of the cluster*"
The cluster that remotely mounts the file systems of the owning cluster

"ciphers": "*Ciphers of the RSA key*"
The list of ciphers of the RSA key. It sets the security mode for communications between the current cluster and the remote cluster. For more information on the list of ciphers, see "mmauth command" on page 96.

"keyDigest": "*SHA digest of the public key*"
The SHA digest of the public key of the remote cluster.

"owningClusterFilesystems": "*Name of the owning cluster file systems.*"
The names of file system of owning cluster and the access permissions to them. Optional.

"OwningFilesystem ":
Owning file system details.

"filesystem ": "*File system on the owning cluster*"
The name of the owning cluster filesystem You can also use the keyword 'all' when addressing all file systems of the owning cluster.

"access": "*Access permissions*"
The access permissions to the owning cluster file system.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example shows how to get the details of the cluster *Cluster1* that mounts file system remotely.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/remotemount/remoteclusters/Cluster1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": 200,
    "message": "..."
  },
  "remoteClusters": [
    {
      "remoteCluster": "Cluster1",
      "ciphers": "[ 'AES128-SHA', 'AES256-SHA' ]",
      "keyDigest": "string",
      "owningClusterFilesystems": "{ {'name':'gpfs0', 'access':'rw'}, {'name':'gpfs1',
'access':'ro'} }"
    }
```

```
  ]
}
```

Related reference

“mmremotecluster command” on page 660
Manages information about remote GPFS clusters.

Remotemount/remoteclusters/{remoteCluster}: PUT

Updates registration of a cluster that can mount one or more file systems of an owning cluster. This API must be run on a cluster that mounts file systems remotely.

Availability

Available on all IBM Spectrum Scale editions.

Note: Only the users with user roles *Administrator* or *Container Operator* have permission to use this REST API endpoint.

Description

The PUT `remotemount/remoteclusters/`*{remoteCluster}* request updates registration of a cluster that can mount one or more file systems of an owning cluster. For more information about the fields in the data structures that are returned, see "mmremotecluster command" on page 660.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/remotemount/remoteclusters/
{remoteCluster}
```

where

remotemount/remoteclusters/*{remoteCluster}*
: Specifies the cluster for which the registration needs to be updated. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 174. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
remoteCluster	Name of the target cluster.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
"remoteClusters":
    {
      "remoteCluster": "Name of the cluster",
      "ciphers": "Ciphers of the RSA key",
      "key": "SHA digest of the public key",
      }
```

"remoteClusters":
: Owning cluster details.

"remoteCluster": "*Name of the cluster*"
The cluster that remotely mounts the file systems of the owning cluster.

"ciphers": "*Ciphers of the RSA key*"
The list of ciphers of the RSA key. It sets the security mode for communications between the current cluster and the remote cluster. For more information on the list of ciphers, see "mmauth command" on page 96.

"key": "*SHA digest of the public key*"
The SHA digest of the public key of the remote cluster.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to update the registration of the cluster *Cluster1*.

Request URL:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
-d '{ \
   "remoteCluster": "Cluster1", \
   "contactNodes": "[%27node1%27, %27node2%27]", \
   "key": ["string"]}'
'https://198.51.100.1:443/scalemgmt/v2/remotemount/remoteclusters/Cluster1'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": ["'mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": ["'(2/3) Linking fileset'']",
        "exitCode": "0",
```

```
        "stderr": "[''EFSSG0740C There are not enough resources available to create
                   a new independent file set.'',  ...]",
        "stdout": "[''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "PUT",
        "url": "/scalemgmt/v2/remotemount/remoteclusters/Cluster1",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

“mmremotecluster command” on page 660
Manages information about remote GPFS clusters.

Remotemount/remoteclusters/{remoteCluster}/access/{owningClusterFilesystem}: POST

Authorizes a cluster to mount a file system remotely. This API must be executed on a cluster that owns file systems that must be mounted remotely.

Availability

Available on all IBM Spectrum Scale editions.

Note: Only the users with user roles *Administrator* or *Container Operator* have permission to use this REST API endpoint.

Description

The POST `remotemount/remoteclusters/`*{remoteCluster}*`/access/`*{owningClusterFilesystem}* request authorizes a cluster to mount a file system remotely. For more information about the fields in the data structures that are returned, see "mmauth command" on page 96.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/remotemount/remoteclusters/
{remoteCluster}/access/{owningClusterFilesystem}
```

where

remotemount/remoteclusters/{remoteCluster}
Specifies the remote cluster where the file system needs to be mounted. Required.

access/{owningClusterFilesystem}
Defines the access privileges to mount the file system. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 175. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.
remoteCluster	Name of the cluster where file system needs to be mounted.	Required.
owningClusterFilesystem	Name of the file system. Specify `all` to authorize the cluster to mount any file system of the owning cluster.	Required.

Request data

```
{
"access": "Access permissions",
 }
```

"access": "*Access permissions*"
The access permissions to the owning cluster file system.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to authorize a cluster to mount a file system remotely.

Request URL:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
-d '{ \
   "access": "rw" \ }'
'https://198.51.100.1:443/scalemgmt/v2/remotemount/remoteclusters/RemoteCluster1/access/
Filesystem1'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": "[''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": "[''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": "[''EFSSG0740C There are not enough resources available to create
                  a new independent file set.'',  ...]",
        "stdout": "[''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "POST",
```

```
        "url": "/scalemgmt/v2/remotemount/remoteclusters/RemoteCluster1/access/Filesystem1",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

“mmauth command” on page 96
Manages secure access to GPFS file systems.

Remotemount/remoteclusters/{remoteCluster}/access/{owningClusterFilesystem}: PUT

Changes the access permission of a cluster to mount a file system remotely. This API must be executed on a cluster that owns file systems that must be mounted remotely.

Availability

Available on all IBM Spectrum Scale editions.

Note: Only the users with user roles *Administrator* or *Container Operator* have permission to use this REST API endpoint.

Description

The PUT `remotemount/remoteclusters/`*{remoteCluster}*`/access/`*{owningClusterFilesystem}* request changes the access permission of a cluster to mount a file system remotely. For more information about the fields in the data structures that are returned, see "mmauth command" on page 96.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/remotemount/remoteclusters/
{remoteCluster}/access/{owningClusterFilesystem}
```

where

remotemount/remoteclusters/{remoteCluster}
Specifies the remote cluster where the file system needs to be mounted. Required.

access/{owningClusterFilesystem}
Defines the access privileges to mount the file system. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 176. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.
remoteCluster	Name of the cluster where file system needs to be mounted.	Required.
owningClusterFilesystem	Name of the file system. Specify `all` to authorize the cluster to mount any file system of the owning cluster.	Required.

Request data

```
{
"access": "Access permissions",
}
```

"access": "*Access permissions*"
The access permissions to the owning cluster file system.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to change the access permission of a cluster to mount a file system remotely.

Request URL:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
-d '{ \
   "access": "rw" \ }'
'https://198.51.100.1:443/scalemgmt/v2/remotemount/remoteclusters/RemoteCluster1/access/
Filesystem1'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": ["'mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": ["'(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": ["'EFSSG0740C There are not enough resources available to create
                  a new independent file set.'',  ...]",
        "stdout": ["'EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
```

```
        "type": "PUT",
        "url": "/scalemgmt/v2/remotemount/remoteclusters/RemoteCluster1/access/Filesystem1",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

“mmauth command” on page 96
Manages secure access to GPFS file systems.

Remotemount/remoteclusters/{remoteCluster}/deny/{owningClusterFilesystem}: DELETE

Denies a cluster from mounting a file system remotely. This API must be executed on a cluster that owns file systems that must be mounted remotely.

Availability

Available on all IBM Spectrum Scale editions.

Note: Only the users with user roles *Administrator* or *Container Operator* have permission to use this REST API endpoint.

Description

The `DELETE remotemount/remoteclusters/{remoteCluster}/deny/{owningClusterFilesystem}` request removes the access permission of a cluster to mount a file system remotely. For more information about the fields in the data structures that are returned, see "mmauth command" on page 96.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/remotemount/remoteclusters/
{remoteCluster}/deny/{owningClusterFilesystem}
```

where

remotemount/remoteclusters/{remoteCluster}
Specifies the remote cluster where the file system needs to be mounted. Required.

deny/{owningClusterFilesystem}
Removes the access privileges to mount the file system. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 177. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
remoteCluster	Name of the cluster where file system needs to be mounted.	Required.
owningClusterFilesystem	Name of the file system. Specify `all` to deny the cluster from mounting any file systems.	Required.

Request data

No request data.

Response data

```
{
     "status": {
       "code":ReturnCode",
       "message":"ReturnMessage"
    },
    jobs: [
        {
           "result":"",
             {
               "commands":"String",
               "progress":"String,
               "exitCode":"Exit code",
               "stderr":"Error",
               "stdout":"String",
             },
           "request":" ",
             {
               "type":"{GET | POST | PUT | DELETE}",
               "url":"URL",
               "data":"",
             }
           "jobId":"ID",
           "submitted":"Time",
           "completed":Time",
           "status":"Job status",
           }
         ],
    }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*'
Array of commands that are run in this job.

"progress":"*String*'
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to remove the access permission of a cluster from mounting a file system remotely.

Request URL:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/remotemount/remoteclusters/RemoteCluster1/deny/
Filesystem1'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": "[''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": "[''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": "[''EFSSG0740C There are not enough resources available to create
                   a new independent file set.'',  ...]",
        "stdout": "[''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "DELETE",
        "url": "/scalemgmt/v2/remotemount/remoteclusters/RemoteCluster1/deny/Filesystem1",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

“mmauth command” on page 96
Manages secure access to GPFS file systems.

Remotemount/remotefilesystems: GET

Lists the remote file systems. This API must be run on a cluster that mounts file systems remotely.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET /remotemount/remotefilesystems` lists the remote file systems. For more information about the fields in the data structures that are returned, see "mmremotefs command" on page 663.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/remotemount/
remotefilesystems
```

where

remotemount/remotefilesystems
Specifies the target of the GET request.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 178. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
owningCluster	Lists remote file systems for a specific owning cluster	Optional

Request data

No request data.

Response data

```
{
  "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   }
   "remoteFilesystems": [
    {
      "remoteFilesystem": "Remote file system name",
      "owningFilesystem": "Owning file system name",
      "owningCluster": "Owning cluster name",
      "remoteMountPath": "Remote mount path of the file system",
      "mountOptions": "Mount options",
      "automount": "Status of the auto mount"
    }
  ]
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"remoteFilesystems":
Remote file system details.

"remoteFilesystem": "*Remote file system name*"
The name of the file system on the remote cluster.

"owningFilesystem": "*Owning file system name*"
The name of the file system on the owning cluster.

"owningCluster": "*Owning cluster name*"
The owning cluster of the remote file system.

"remoteMountPath": "*Remote mount path of the file system*"
The remote mount path of the remote file system.

"mountOptions": "*Mount options*"
The mount options of the remote file system.

"automount": "*Status of the auto mount*"
The automount status of the remote file system.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example shows how to get the list of remote file systems that belong to the owning cluster *Cluster1*.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/remotemount/remotefilesystems?owningCluster=Cluster1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": 200,
    "message": "..."
  },
   "remoteFilesystems": [
    {
      "remoteFilesystem": "filesystem1",
      "owningFilesystem": "Owningfileystem1",
      "owningCluster": "Cluster1",
      "remoteMountPath": "string",
      "mountOptions": "string",
      "automount": "string"
    }
  ]
```

Related reference
“mmremotecluster command” on page 660
Manages information about remote GPFS clusters.

Remotemount/remotefilesystems: POST

Creates a remote file system. This API must be run on a cluster that mounts file systems remotely.

Availability

Available on all IBM Spectrum Scale editions.

Note: Only the users with user roles *Administrator* or *Container Operator* have permission to use this REST API endpoint.

Description

The POST `/remotemount/remotefilesystems` creates a remote file system. For more information about the fields in the data structures that are returned, see “mmremotefs command” on page 663.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/remotemount/
remotefilesystems
```

where

remotemount/remotefilesystems
Specifies the target element that is going to be created.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 179. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
"remoteFilesystems":
    {
      "remoteFilesystem": "Remote file system name",
      "owningFilesystem": "Owning file system name",
      "owningCluster": "Owning cluster name",
      "remoteMountPath": "Remote mount path of the file system",
      "mountOptions": "Mount options",
      "automount": "Status of the auto mount"
    }
```

"remoteFilesystems":
Remote file system details.

"remoteFilesystem": "*Remote file system name*"
The name of the remote file system on the remote cluster.

"owningFilesystem": "*Owning file system name*"
The name of the file system on the owning cluster.

"owningCluster": "*Owning cluster name*"
The owning cluster of the remote file system.

"remoteMountPath": "*Remote mount path of the file system*"
The remote mount path of the remote file system.

"mountOptions": "*Mount options*"
The mount options of the remote file system.

"automount": "*Status of the auto mount*"
The automount status of the remote file system.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
        ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to create a remote file system.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
-d '{ \
   "remoteFilesystem": "RemoteFilesystem1", \
   "owningFilesystem": "owningfilesystem1", \
   "owningCluster": "Cluster1", \
   "remoteMountPath": "string", \
   "mountOptions": "string", \
   "automount": "string" \
'https://198.51.100.1:443/scalemgmt/v2/remotemount/remotefilesystems'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
```

```
      "result": {
        "commands": ['''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": ['''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": ['''EFSSG0740C There are not enough resources available to create
                    a new independent file set.'',  ...]",
        "stdout": ['''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "POST",
        "url": "/scalemgmt/v2/remotemount/remotefilesystems",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

Related reference

“mmremotecluster command” on page 660
Manages information about remote GPFS clusters.

Remotemount/remotefilesystems/{remoteFilesystem}: DELETE

Deletes a remote file system. This API must be run on a cluster that mounts file systems remotely.

Availability

Available on all IBM Spectrum Scale editions.

Note: Only the users with user roles *Administrator* or *Container Operator* have permission to use this REST API endpoint.

Description

The `DELETE /remotemount/remotefilesystems/{remoteFilesystem}` deletes the details of a remote file system. For more information about the fields in the data structures that are returned, see "mmremotefs command" on page 663.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/remotemount/
remotefilesystems/{remoteFilesystem}
```

where

remotemount/remotefilesystems/{remoteFilesystem}
Specifies the file system to be deleted.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 180. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
remoteFilesystem	Name of the remote file system to be deleted.	Required.

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
```

```
            "stdout":"String",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted":"Time",
        "completed":Time",
        "status":"Job status",
        }
      ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to delete the remote file system *RemoteFilesystem1*.

Request data:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/remotemount/remotefilesystems/RemoteFilesystem1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": "[''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": "[''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": "[''EFSSG0740C There are not enough resources available to create
                  a new independent file set.'',  ...]",
        "stdout": "[''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "DELETE",
        "url": "/scalemgmt/v2/remotemount/remotefilesystems/RemoteFilesystem1",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

Related reference

“mmremotecluster command” on page 660
Manages information about remote GPFS clusters.

Remotemount/remotefilesystems/{remoteFilesystem}: GET

Lists the details of a remote file system. This API must be run on a cluster that mounts file systems remotely.

Availability

Available on all IBM Spectrum Scale editions.

Description

The GET `/remotemount/remotefilesystems/{remoteFilesystem}` gets the details of a remote file system. For more information about the fields in the data structures that are returned, see "mmremotefs command" on page 663.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/remotemount/
remotefilesystems/{remoteFilesystem}
```

where

remotemount/remotefilesystems/{remoteFilesystem}
Specifies the target of the GET request.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 181. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
remoteFilesystem	Name of the remote file system.	Required.

Request data

No request data.

Response data

```
{
  "status": {
      "code":ReturnCode,
      "message":"ReturnMessage"
   }
   "remoteFilesystems": [
    {
      "remoteFilesystem": "Remote file system name",
      "owningFilesystem": "Owning file system name",
      "owningCluster": "Owning cluster name",
      "remoteMountPath": "Remote mount path of the file system",
      "mountOptions": "Mount options",
      "automount": "Status of the auto mount"
    }
  ]
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"remoteFilesystems":
Remote file system details.

"remoteFilesystem": "*Remote file system name*"
The name of the remote file system on the remote cluster.

"owningFilesystem": "*Owning file system name*"
The name of the file system on the owning cluster.

"owningCluster": "*Owning cluster name*"
The owning cluster of the remote file system.

"remoteMountPath": "*Remote mount path of the file system*"
The remote mount path of the remote file system.

"mountOptions": "*Mount options*"
The mount options of the remote file system.

"automount": "*Status of the auto mount*"
The automount status of the remote file system.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

Examples

The following example shows how to get the details of the file system *RemoteFilesystem1*.

Request data:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/remotemount/remotefilesystems/RemoteFilesystem1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": 200,
    "message": "..."
  },
   "remoteFilesystems": [
    {
      "remoteFilesystem": "RemoteFilesystem1",
      "owningFilesystem": "Owningfileystem1",
      "owningCluster": "Cluster1",
      "remoteMountPath": "string",
      "mountOptions": "string",
      "automount": "string"
    }
  ]
```

Related reference
“mmremotecluster command” on page 660
Manages information about remote GPFS clusters.

Remotemount/remotefilesystems/{remoteFilesystem}: PUT

Updates an existing remote file system. This API must be run on a cluster that mounts file systems remotely.

Availability

Available on all IBM Spectrum Scale editions.

Note: Only the users with user roles *Administrator* or *Container Operator* have permission to use this REST API endpoint.

Description

The PUT `/remotemount/remotefilesystems/`*{remoteFilesystem}* updates the details of a remote file system. For more information about the fields in the data structures that are returned, see "mmremotefs command" on page 663.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/remotemount/
remotefilesystems/{remoteFilesystem}
```

where

remotemount/remotefilesystems/{remoteFilesystem}
Specifies the remote file system to be updated.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 182. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
remoteFilesystem	Name of the remote file system to be updated.	Required.
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
"remoteFilesystems": [
   {
     "remoteFilesystem": "Remote file system name",
     "owningFilesystem": "Owning file system name",
     "owningCluster": "Owning cluster name",
     "remoteMountPath": "Remote mount path of the file system",
     "mountOptions": "Mount options",
```

```
      "automount": "Status of the auto mount"
    }
```

"remoteFilesystems":
Remote file system details.

"remoteFilesystem": "*Remote file system name*"
The name of the remote file system on the remote cluster.

"owningFilesystem": "*Owning file system name*"
The name of the file system on the owning cluster.

"owningCluster": "*Owning cluster name*"
The owning cluster of the remote file system.

"remoteMountPath": "*Remote mount path of the file system*"
The remote mount path of the remote file system.

"mountOptions": "*Mount options*"
The mount options of the remote file system.

"automount": "*Status of the auto mount*"
The automount status of the remote file system.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, and nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example shows how to update the remote file system *RemoteFilesystem1*.

Request data:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
-d '{ \
   "remoteFilesystem": "RemoteFilesystem1", \
   "owningFilesystem": "owningfilesystem1", \
   "owningCluster": "Cluster1", \
   "remoteMountPath": "string", \
   "mountOptions": "string", \
   "automount": "string" \
'https://198.51.100.1:443/scalemgmt/v2/remotemount/remotefilesystems/RemoteFilesystem1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": "[''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": "[''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": "[''EFSSG0740C There are not enough resources available to create
                   a new independent file set.'',  ...]",
        "stdout": "[''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "PUT",
        "url": "/scalemgmt/v2/remotemount/remotefilesystems/RemoteFilesystem1",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

Related reference

“mmremotecluster command” on page 660
Manages information about remote GPFS clusters.

SMB/shares: GET

Gets information about SMB shares.

Availability

Available on all IBM Spectrum Scale editions.

Description

The GET `smb/shares` request gets information about SMB shares. For more information about the fields in the data structures that are returned, see “mmsmb command” on page 709.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/smb/shares
```

where

`smb/shares`
Specifies SMB shares as the resource. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 183. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Request data

No request data.

Response data

The following list of attributes are available in the response data:

```
{
"status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "paging":
    {
    "next": "URL"
```

```
    },
  "SmbShare":
    {
         "shareName": "Share name"
         "filesystemName": File system name
         "filesetName": Fileset Name
         "path": "Path",
      ),
      "smbOptions": "Options"
      (
         "browseable":""Browseable"
         "smbEncrypt": "auto | default | mandatory | disabled"
         "adminUsers": "Users"
         "comment": "Comments"
         "cscPolicy": "Client-side caching policy"
         "fileIdAlgorithm": "fsname | hostname | fsnamenodirs |
    fsnamenorootdir"
         "gpfsLeases": "yes | no"
         "gpfsRecalls": "yes | no"
         "gpfsShareModes": "yes | no"
         "gpfsSyncIo": "yes | no"
         "hideUnreadable": "yes | no"
         "opLocks": "yes | no"
         "posixLocking": "yes | no"
         "readOnly": "yes | no"
         "syncOpsOnClose": "yes | no"
         "hideDotFiles": "yes | no"
         }
```

The details of the parameters are given in the following list:

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"Smbshare":
An array of information about SMB shares addresses. The array contains elements that describe the SMB shares. For more information about the fields in this structure, see the links at the end of this topic.

"shareName": "*String*"
Name of the SMB share.

"filesystemName": "*String*"
The file system to which the SMB share belongs.

"filesetName": "*String*"
The fileset to which the SMB share belongs.

"path": "*String*"
The path for which SMB share is created.

"smbOptions":
An array of information about SMB shares addresses. The array contains elements that describe the SMB shares. For more information about the fields in this structure, see the links at the end of this topic.

"browseable":""*Browseable*"
If the value is set as yes, the export is shown in the Windows Explorer browser while browsing the file server.

"smbencrypt": "{auto | default | mandatory | disabled}"
This option controls whether the remote client is allowed or required to use SMB encryption. This option controls whether the remote client is allowed or required to use SMB encryption.

"adminUsers": "*Users*"
Using this option, administrative users can be defined in the format of admin users=user1;user2, ..usern. The users must be domain users.

"comment": "*Comments*"
User-defined text.

"cscPolicy": "{manual | disable | documents | programs}"
The client-side caching policy specifies how the clients that are capable of offline caching caches the files in the share.

"fileIdAlgorithm": "{fsname | hostname | fsnamenodirs | fsnamenorootdir}"
This option allows to control the level of enforced data integrity. If the data integrity is ensured on the application level, it can be beneficial in cluster environments to reduce the level of enforced integrity for performance reasons.

"gpfsLeases": "yes | no"
These are cross protocol oplocks (opportunistic locks). That is, an SMB client can lock a file that provides the user improved performance while reading or writing to the file because no other user read or write to this file. If the value is set as yes, clients accessing the file over the other protocols can break the lock of an SMB client and the user gets informed when another user is accessing the same file at the same time.

"gpfsRecalls": "{yes | no}"
If the value is set as yes, files that are migrated from disk recalled on access. If recalls = no, files are not recalled on access and the client receives.

"gpfsShareModes": "{yes | no}"
An application can set share modes. If you set gpfs:sharemodes = yes, the share modes that are specified by the application is respected by all protocols and not only by the SMB protocol. If you set gpfs:sharemodes = no, then the share modes that are specified by the application is only respected by the SMB protocol.

"gpfsSyncIo": "{yes | no}"
If the value is set as yes, the files in an export for which the setting is enabled, are opened with the OSYNC flag. Accessing a file is faster if gpfs:syncio is set to yes. Performance for certain workloads can be improved when SMB accesses the file with the OSYNC flag set. For example, updating only small blocks in a large file as observed with database applications. The underlying GPFS behavior is then changed to not read a complete block, if there is only a small update to it.

"hideUnreadable": "{yes | no}"
If the value is set as yes, all files and directories that the user has no permission to read is hidden from directory listings in the export. The *hideunreadable=yes* option is also known as access-based enumeration. When users are listing (enumerating) the directories and files within the export, they see only the files and directories that they have read access to.

"opLocks": "{yes | no}"
If the value is set as yes, a client might request an opportunistic lock (oplock) from an SMB server when it opens a file. If the server grants the request, the client can cache large chunks of the file without informing the server what it is doing with the cached chunks until the task is completed. Caching large chunks of a file saves a lot of network I/O round-trip time and enhances performance.

"posixLocking": "{yes | no}"
If the value is set as yes, it is tested if a byte–range (fcntl) lock is already present on the requested portion of the file before granting a byte–range lock to an SMB client. For improved performance on SMB–only shares this option can be disabled. Disabling locking on cross–protocol shares can result in data integrity issues when clients concurrently set locks on a file through multiple protocols, for example, SMB and NFS.

"readOnly": "{yes | no}"
If the value is set as yes, files cannot be modified or created on this export independent of the ACLs.

"syncOpsOnClose": "{yes | no}"
This option ensures that the file system synchronizes data to the disk each time a file is closed after writing. The written data is flushed to the disk.

"hideDotFiles": "{yes | no}"
If the value is set to yes, files starting with "." are hidden.

Examples

The following example gets information about the SMB shares that are configured in the system.

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/smb/shares
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "smbShares" : [ {
    "config" : {
      "shareName" : "share01"
    }
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully"
  }
}
```

Using the field parameter ":all:" returns entire details of the SMB shares. For example:

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/smb/shares/smbShare1?lastId=1001"
  },
  "SmbShares": [
    {
        "shareName": "smbShare1",
        "filesystemName": "gpfs0",
        "filesetName": "fset1",
        "path": "/mnt/gpfs0/fset1"
      },
      "smbOptions": {
        "browseable": "yes",
        "smbEncrypt": "auto",
        "adminUsers": "admin",
        "comment": "This is a comment",
        "cscPolicy": "manual",
        "fileIdAlgorithm": "fsname",
        "gpfsLeases": "yes",
        "gpfsRecalls": "yes",
        "gpfsShareModes": "yes",
        "gpfsSyncIo": "yes",
        "hideUnreadable": "yes",
        "opLocks": "yes",
        "posixLocking": "yes",
        "readOnly": "yes",
        "syncOpsOnClose": "yes"
        "hideDotFiles": "yes"
      }
    }
  ]
}
```

Related reference

"mmsmb command" on page 709
Administers SMB shares, export ACLs, and global configuration.

SMB/shares/{shareName}: GET

Gets information about a specific SMB share.

Availability

Available on all IBM Spectrum Scale editions.

Description

The GET `smb/shares/shareName` request gets information about the specified SMB share. For more information about the fields in the data structures that are returned, see “mmsmb command” on page 709.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/smb/shares/shareName
```

where

smb/shares
Specifies the SMB share as the resource. Required.

shareName
Specifies the SMB share about which you need to get the details. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 184. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
shareName	Name of the SMB share.	Required.

Request data

No request data.

Response data

The following list of attributes are available in the response data:

```
{
"status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "paging":
    {
```

```
    "next": "URL"
    },
  "SmbShare":
    {
         "shareName": "Share name"
         "filesystemName": File system name
         "filesetName": Fileset Name
         "path": "Path",
      ),
      "smbOptions": "Options"
      (
         "browseable":""Browseable"
         "smbEncrypt": "auto | default | mandatory | disabled"
         "adminUsers": "Users"
         "comment": "Comments"
         "cscPolicy": "Client-side caching policy"
         "fileIdAlgorithm": "fsname | hostname | fsnamenodirs |
    fsnamenorootdir"
         "gpfsLeases": "yes | no"
         "gpfsRecalls": "yes | no"
         "gpfsShareModes": "yes | no"
         "gpfsSyncIo": "yes | no"
         "hideUnreadable": "yes | no"
         "opLocks": "yes | no"
         "posixLocking": "yes | no"
         "readOnly": "yes | no"
         "syncOpsOnClose": "yes | no"
         "hideDotFiles": "yes | no"
         }
```

The details of the parameters are given in the following list:

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"SmbShares":
An array of information about SMB shares addresses. The array contains elements that describe the SMB shares. For more information about the fields in this structure, see the links at the end of this topic.

"shareName": "*String*"
Name of the SMB share.

"filesystemName": "*String*"
The file system to which the SMB share belongs.

"filesetName": "*String*"
The fileset to which the SMB share belongs.

"path": "*String*"
The path for which SMB share is created.

"smbOptions":
An array of information about SMB shares addresses. The array contains elements that describe the SMB shares. For more information about the fields in this structure, see the links at the end of this topic.

"browseable":""*Browseable*"
If the value is set as yes, the export is shown in the Windows Explorer browser while browsing the file server.

"smbencrypt": "{auto | default | mandatory | disabled}"
This option controls whether the remote client is allowed or required to use SMB encryption. This option controls whether the remote client is allowed or required to use SMB encryption.

"adminUsers": "*Users*"
Using this option, administrative users can be defined in the format of admin users=user1;user2, ..usern. The users must be domain users.

"comment": "*Comments*"
User-defined text.

"cscPolicy": "{manual | disable | documents | programs}"
The client-side caching policy specifies how the clients that are capable of offline caching caches the files in the share.

"fileIdAlgorithm": "{fsname | hostname | fsnamenodirs | fsnamenorootdir}"
This option allows to control the level of enforced data integrity. If the data integrity is ensured on the application level, it can be beneficial in cluster environments to reduce the level of enforced integrity for performance reasons.

"gpfsLeases": "yes | no"
These are cross protocol oplocks (opportunistic locks). That is, an SMB client can lock a file that provides the user improved performance while reading or writing to the file because no other user read or write to this file. If the value is set as yes, clients accessing the file over the other protocols can break the lock of an SMB client and the user gets informed when another user is accessing the same file at the same time.

"gpfsRecalls": "{yes | no}"
If the value is set as yes, files that are migrated from disk recalled on access. If recalls = no, files are not recalled on access and the client receives.

"gpfsShareModes": "{yes | no}"
An application can set share modes. If you set gpfs:sharemodes = yes, the share modes that are specified by the application is respected by all protocols and not only by the SMB protocol. If you set gpfs:sharemodes = no, then the share modes that are specified by the application is only respected by the SMB protocol.

"gpfsSyncIo": "{yes | no}"
If the value is set as yes, the files in an export for which the setting is enabled, are opened with the OSYNC flag. Accessing a file is faster if gpfs:syncio is set to yes. Performance for certain workloads can be improved when SMB accesses the file with the OSYNC flag set. For example, updating only small blocks in a large file as observed with database applications. The underlying GPFS behavior is then changed to not read a complete block, if there is only a small update to it.

"hideUnreadable": "{yes | no}"
If the value is set as yes, all files and directories that the user has no permission to read is hidden from directory listings in the export. The *hideunreadable=yes* option is also known as access-based enumeration. When users are listing (enumerating) the directories and files within the export, they see only the files and directories that they have read access to.

"opLocks": "{yes | no}"
If the value is set as yes, a client might request an opportunistic lock (oplock) from an SMB server when it opens a file. If the server grants the request, the client can cache large chunks of the file without informing the server what it is doing with the cached chunks until the task is completed. Caching large chunks of a file saves a lot of network I/O round-trip time and enhances performance.

"posixLocking": "{yes | no}"
If the value is set as yes, it is tested if a byte–range (fcntl) lock is already present on the requested portion of the file before granting a byte–range lock to an SMB client. For improved performance on SMB–only shares this option can be disabled. Disabling locking on cross–protocol shares can result in data integrity issues when clients concurrently set locks on a file through multiple protocols, for example, SMB and NFS.

"readOnly": "{yes | no}"
If the value is set as yes, files cannot be modified or created on this export independent of the ACLs.

"syncOpsOnClose": "{yes | no}"
This option ensures that the file system synchronizes data to the disk each time a file is closed after writing. The written data is flushed to the disk.

"hideDotFiles": "{yes | no}"
If the value is set to yes, files starting with "." are hidden.

Examples

The following example gets information about the SMB share `share1`.

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/smb/shares/smbShare1
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/smb/shares/smbShare1?lastId=1001"
  },
  "SmbShares": [
    {
        "shareName": "smbShare1",
        "filesystemName": "gpfs0",
        "filesetName": "fset1",
        "path": "/mnt/gpfs0/fset1"
      },
      "smbOptions": {
        "browseable": "yes",
        "smbEncrypt": "auto",
        "adminUsers": "admin",
        "comment": "This is a comment",
        "cscPolicy": "manual",
        "fileIdAlgorithm": "fsname",
        "gpfsLeases": "yes",
        "gpfsRecalls": "yes",
        "gpfsShareModes": "yes",
        "gpfsSyncIo": "yes",
        "hideUnreadable": "yes",
        "opLocks": "yes",
        "posixLocking": "yes",
        "readOnly": "yes",
        "syncOpsOnClose": "yes"
        "hideDotFiles": "yes"
      }
    }
  ]
}
```

Related reference

“mmsmb command” on page 709
Administers SMB shares, export ACLs, and global configuration.

SMB/shares: POST

Creates a new SMB share.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `smb/shares` request creates a new SMB share. For more information about the fields in the data structures that are returned, see “mmsmb command” on page 709.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/smb/shares/shareName
/scalemgmt/v2/smb/shares
```

where

`smb/shares`
Specifies SMB share as the target. Required.

Request headers

```
Accept: application/json
```

Request data

The following list of attributes are available in the response data:

```
{        "shareName": "ShareName"
         "path": "Path",
         "smbOptions"
          {
         "browseable":""Browseable"
         "smbEncrypt": "{auto | default | mandatory | disabled}"
         "adminUsers": "Users"
         "comment": "Comments"
         "cscPolicy": "Client-side caching policy"
         "fileIdAlgorithm": "fsname | hostname | fsnamenodirs |
    fsnamenorootdir"
         "gpfsLeases": "{yes | no}"
         "gpfsRecalls": "{yes | no}"
         "gpfsShareModes,": "{yes | no}"
         "gpfsSyncIo": "{yes | no}"
         "hideUnreadable": "{yes | no}"
         "opLocks": "{yes | no}"
         "posixLocking": "{yes | no}"
         "readOnly": "{yes | no}"
         "syncOpsOnClose": "{yes | no}"
         "hideDotFiles": "yes | no"
         }
}
```

The details of the parameters are given in the following list:

`config":`
Configuration details of SMB share.

`"shareName": "`*`String`*`"`
Name of the SMB share.

`"filesystemName": "`*`String`*`"`
The file system to which the SMB share belongs.

"filesetName": "*String*"
The fileset to which the SMB share belongs.

"path": "*String*"
The path for which SMB share is created.

"smbOptions":
An array of information about SMB shares addresses. The array contains elements that describe the SMB shares. For more information about the fields in this structure, see the links at the end of this topic.

"smbOptions":
An array of information about SMB shares addresses. The array contains elements that describe the SMB shares. For more information about the fields in this structure, see the links at the end of this topic.

"browseable":"*Browseable*"
If the value is set as yes, the export is shown in the Windows Explorer browser while browsing the file server.

"smbEncrypt": "{auto | default | mandatory | disabled}"
This option controls whether the remote client is allowed or required to use SMB encryption. This option controls whether the remote client is allowed or required to use SMB encryption.

"adminUsers": "*Users*"
Using this option, administrative users can be defined in the format of admin users=user1;user2, ..usern. The users must be domain users.

"comment": "*Comments*"
User-defined text.

"cscPolicy": "{manual | disable | documents | programs}"
The client-side caching policy specifies how the clients that are capable of offline caching caches the files in the share.

"fileIdAlgorithm": "{fsname | hostname | fsnamenodirs | fsnamenorootdir}"
This option allows to control the level of enforced data integrity. If the data integrity is ensured on the application level, it can be beneficial in cluster environments to reduce the level of enforced integrity for performance reasons.

"gpfsLeases": "yes | no"
These are cross protocol oplocks (opportunistic locks). That is, an SMB client can lock a file that provides the user improved performance while reading or writing to the file because no other user read or write to this file. If the value is set as yes, clients accessing the file over the other protocols can break the lock of an SMB client and the user gets informed when another user is accessing the same file at the same time.

"gpfsRecalls": "{yes | no}"
If the value is set as yes, files that are migrated from disk recalled on access. If recalls = no, files are not recalled on access and the client receives.

"gpfsShareModes": "{yes | no}"
An application can set share modes. If you set gpfs:sharemodes = yes, the share modes that are specified by the application is respected by all protocols and not only by the SMB protocol. If you set gpfs:sharemodes = no, then the share modes that are specified by the application is only respected by the SMB protocol.

"gpfsSyncIo": "{yes | no}"
If the value is set as yes, the files in an export for which the setting is enabled, are opened with the OSYNC flag. Accessing a file is faster if gpfs:syncio is set to yes. Performance for certain workloads can be improved when SMB accesses the file with the OSYNC flag set. For example, updating only small blocks in a large file as observed with database applications. The underlying GPFS behavior is then changed to not read a complete block, if there is only a small update to it.

"hideUnreadable": "{yes | no}"
If the value is set as yes, all files and directories that the user has no permission to read is hidden from directory listings in the export. The *hideunreadable=yes* option is also known as access-based enumeration. When users are listing (enumerating) the directories and files within the export, they see only the files and directories that they have read access to.

"opLocks": "{yes | no}"
If the value is set as yes, a client might request an opportunistic lock (oplock) from an SMB server when it opens a file. If the server grants the request, the client can cache large chunks of the file without informing the server what it is doing with the cached chunks until the task is completed. Caching large chunks of a file saves a lot of network I/O round-trip time and enhances performance.

"posixLocking": "{yes | no}"
If the value is set as yes, it is tested if a byte–range (fcntl) lock is already present on the requested portion of the file before granting a byte–range lock to an SMB client. For improved performance on SMB–only shares this option can be disabled. Disabling locking on cross–protocol shares can result in data integrity issues when clients concurrently set locks on a file through multiple protocols, for example, SMB and NFS.

"readOnly": "{yes | no}"
If the value is set as yes, files cannot be modified or created on this export independent of the ACLs.

"syncOpsOnClose": "{yes | no}"
This option ensures that the file system synchronizes data to the disk each time a file is closed after writing. The written data is flushed to the disk.

"hideDotFiles": "{yes | no}"
If the value is set to yes, files starting with "." are hidden.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example creates a new SMB share.

Request data:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
-d '{
  "shareName": "smbShare",
  "path": "/mnt/gpfs0/fset1",
  "smbOptions": {
    "browseable": "yes",
    "smbEncrypt": "auto",
    "adminUsers": "admin",
    "comment": "This is a comment",
    "cscPolicy": "manual",
    "fileIdAlgorithm": "fsname",
    "gpfsLeases": "yes",
    "gpfsRecalls": "yes",
    "gpfsShareModes": "yes",
    "gpfsSyncIo": "yes",
    "hideUnreadable": "yes",
    "opLocks": "yes",
    "posixLocking": "yes",
    "readOnly": "yes",
    "syncOpsOnClose": "yes",
    "hideDotFiles": "yes"
```

```
  }
}
}' "https://198.51.100.1:443/scalemgmt/v2/smb/shares"
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {},
      "request": {
        "type": "POST",
        "url": "/scalemgmt/v2/smb/shares",
        "data": "{
            "shareName": "smbShare",
            "path": "/mnt/gpfs0/fset1",
            "smbOptions": {
            "browseable": "yes",
            "smbEncrypt": "auto",
            "adminUsers": "admin",
            "comment": "This is a comment",
            "cscPolicy": "manual",
            "fileIdAlgorithm": "fsname",
            "gpfsLeases": "yes",
            "gpfsRecalls": "yes",
            "gpfsShareModes": "yes",
            "gpfsSyncIo": "yes",
            "hideUnreadable": "yes",
            "opLocks": "yes",
            "posixLocking": "yes",
            "readOnly": "yes",
            "syncOpsOnClose": "yes",
            "hideDotFiles": "yes"
        },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

Related reference

“mmsmb command” on page 709
Administers SMB shares, export ACLs, and global configuration.

SMB/shares/{shareName}: PUT

Modifies an existing SMB share.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `smb/shares/shareName` request modifying an existing SMB share. For more information about the fields in the data structures that are returned, see “mmsmb command” on page 709.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/smb/shares/shareName
```

where

`smb/shares`
Specifies SMB share as the resource. Required.

shareName
Specifies the SMB share to be modified. Required.

Request headers

```
Accept: application/json
```

Request data

The following list of attributes are available in the response data:

```
{        "shareName": "ShareName"
         "path": "Path",
         "smbOptions"
          {
         "browseable":""Browseable"
         "smbEncrypt": "{auto | default | mandatory | disabled}"
         "adminUsers": "Users"
         "comment": "Comments"
         "cscPolicy": "Client-side caching policy"
         "fileIdAlgorithm": "fsname | hostname | fsnamenodirs |
    fsnamenorootdir"
         "gpfsLeases": "{yes | no}"
         "gpfsRecalls": "{yes | no}"
         "gpfsShareModes,": "{yes | no}"
         "gpfsSyncIo": "{yes | no}"
         "hideUnreadable": "{yes | no}"
         "opLocks": "{yes | no}"
         "posixLocking": "{yes | no}"
         "readOnly": "{yes | no}"
         "syncOpsOnClose": "{yes | no}"
         "hideDotFiles": "yes | no"
         }
}
```

The details of the parameters are given in the following list:

"smbOptions":
An array of information about SMB shares addresses. The array contains elements that describe the SMB shares. For more information about the fields in this structure, see the links at the end of this topic.

"smbOptions":

An array of information about SMB shares addresses. The array contains elements that describe the SMB shares. For more information about the fields in this structure, see the links at the end of this topic.

"browseable":""*Browseable*"

If the value is set as yes, the export is shown in the Windows Explorer browser while browsing the file server.

"smbEncrypt": "{auto | default | mandatory | disabled}"

This option controls whether the remote client is allowed or required to use SMB encryption. This option controls whether the remote client is allowed or required to use SMB encryption.

"adminUsers": "*Users*"

Using this option, administrative users can be defined in the format of admin users=user1;user2, ..usern. The users must be domain users.

"comment": "*Comments*"

User-defined text.

"cscPolicy": "{manual | disable | documents | programs}"

The client-side caching policy specifies how the clients that are capable of offline caching caches the files in the share.

"fileIdAlgorithm": "{fsname | hostname | fsnamenodirs | fsnamenorootdir}"

This option allows to control the level of enforced data integrity. If the data integrity is ensured on the application level, it can be beneficial in cluster environments to reduce the level of enforced integrity for performance reasons.

"gpfsLeases": "yes | no"

These are cross protocol oplocks (opportunistic locks). That is, an SMB client can lock a file that provides the user improved performance while reading or writing to the file because no other user read or write to this file. If the value is set as yes, clients accessing the file over the other protocols can break the lock of an SMB client and the user gets informed when another user is accessing the same file at the same time.

"gpfsRecalls": "{yes | no}"

If the value is set as yes, files that are migrated from disk recalled on access. If recalls = no, files are not recalled on access and the client receives.

"gpfsShareModes": "{yes | no}"

An application can set share modes. If you set gpfs:sharemodes = yes, the share modes that are specified by the application is respected by all protocols and not only by the SMB protocol. If you set gpfs:sharemodes = no, then the share modes that are specified by the application is only respected by the SMB protocol.

"gpfsSyncIo": "{yes | no}"

If the value is set as yes, the files in an export for which the setting is enabled, are opened with the OSYNC flag. Accessing a file is faster if gpfs:syncio is set to yes. Performance for certain workloads can be improved when SMB accesses the file with the OSYNC flag set. For example, updating only small blocks in a large file as observed with database applications. The underlying GPFS behavior is then changed to not read a complete block, if there is only a small update to it.

"hideUnreadable": "{yes | no}"

If the value is set as yes, all files and directories that the user has no permission to read is hidden from directory listings in the export. The *hideunreadable=yes* option is also known as access-based enumeration. When users are listing (enumerating) the directories and files within the export, they see only the files and directories that they have read access to.

"opLocks": "{yes | no}"

If the value is set as yes, a client might request an opportunistic lock (oplock) from an SMB server when it opens a file. If the server grants the request, the client can cache large chunks of the file without informing the server what it is doing with the cached chunks until the task

is completed. Caching large chunks of a file saves a lot of network I/O round-trip time and enhances performance.

"posixLocking": "{yes | no}"
If the value is set as yes, it is tested if a byte–range (fcntl) lock is already present on the requested portion of the file before granting a byte–range lock to an SMB client. For improved performance on SMB–only shares this option can be disabled. Disabling locking on cross–protocol shares can result in data integrity issues when clients concurrently set locks on a file through multiple protocols, for example, SMB and NFS.

"readOnly": "{yes | no}"
If the value is set as yes, files cannot be modified or created on this export independent of the ACLs.

"syncOpsOnClose": "{yes | no}"
This option ensures that the file system synchronizes data to the disk each time a file is closed after writing. The written data is flushed to the disk.

"hideDotFiles": "{yes | no}"
If the value is set to yes, files starting with "." are hidden.

"removeSmbOptions": ["*List of SMB options to be removed*"]
Specifies the SMB options to be removed.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example modifies the SMB share `smbShare1`.

Request data:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json' -d '{
  "shareName": "smbShare",
  "path": "/mnt/gpfs0/fset1",
  "smbOptions": {
    "browseable": "yes",
    "smbEncrypt": "auto",
    "adminUsers": "admin",
    "comment": "This is a comment",
    "cscPolicy": "manual",
    "fileIdAlgorithm": "fsname",
    "gpfsLeases": "yes",
    "gpfsRecalls": "yes",
    "gpfsShareModes": "yes",
    "gpfsSyncIo": "yes",
    "hideUnreadable": "yes",
    "opLocks": "yes",
    "posixLocking": "yes",
    "readOnly": "yes",
    "syncOpsOnClose": "yes",
    "hideDotFiles": "yes"
  }
}' "https://198.51.100.1:443/scalemgmt/v2/smb/shares/smbShare1"
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {},
      "request": {
        "type": "PUT",
        "url": "/scalemgmt/v2/smb/shares/smbShare1",
        "data": "{
            "shareName": "smbShare",
            "path": "/mnt/gpfs0/fset1",
            "smbOptions": {
            "browseable": "yes",
            "smbEncrypt": "auto",
            "adminUsers": "admin",
            "comment": "This is a comment",
            "cscPolicy": "manual",
            "fileIdAlgorithm": "fsname",
            "gpfsLeases": "yes",
            "gpfsRecalls": "yes",
            "gpfsShareModes": "yes",
            "gpfsSyncIo": "yes",
            "hideUnreadable": "yes",
            "opLocks": "yes",
            "posixLocking": "yes",
            "readOnly": "yes",
            "syncOpsOnClose": "yes",
            "hideDotFiles": "yes"
        },
         "removeSmbOptions": [
         "string"
  ]}"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

Related reference

“mmsmb command” on page 709
Administers SMB shares, export ACLs, and global configuration.

SMB/shares/{shareName}: DELETE

Deletes an SMB share.

Availability

Available on all IBM Spectrum Scale editions.

Description

The DELETE smb/shares/shareName command deletes the specified SMB share. For more information, see “mmsmb command” on page 709.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/smb/shares/shareName
```

where:

smb/shares
Specifies the SMB share as the resource. Required.

shareName
Specifies the SMB share to be deleted. Required.

Request headers

```
Accept: application/json
```

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example deletes the SMB share `share1`.

Request data:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https:///198.51.100.1:443/scalemgmt/v2/smb/shares/share1'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
"jobs" : [ {
"jobId" : 1000000000008,
"status" : "RUNNING",
"submitted" : "2017-08-24 04:45:34,034",
"completed" : "N/A",
"request" : {
"type" : "DELETE",
```

```
"url" : "/scalemgmt/v2/smb/shares/share1"
},
"result" : { },
"pids" : [ 15084 ]
} ],
"status" : {
"code" : 200,
"message" : "The request finished successfully"
}
}
```

Related reference

“mmsmb command” on page 709
Administers SMB shares, export ACLs, and global configuration.

SMB/shares/{shareName}/acl: DELETE

Deletes access control list (ACL) applicable to an SMB share. When you delete the ACL, the ACL is not actually deleted but it basically replaces the current ACL with one that grants Everyone full access.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `DELETE smb/shares/shareName/acl` command deletes the ACLs of the specified SMB share. For more information, see "mmsmb command" on page 709.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/smb/shares/shareName/acl
```

where:

smb/shares
Specifies the SMB share as the resource. Required.

shareName/acl
Specifies the SMB share of which the ACL must be deleted. Required.

Request headers

```
Accept: application/json
```

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
        ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example deletes ACL of the SMB share `share1`.

Request data:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https:///198.51.100.1:443/scalemgmt/v2/smb/shares/share1/acl'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
"jobs" : [ {
"jobId" : 1000000000008,
"status" : "RUNNING",
"submitted" : "2017-08-24 04:45:34,034",
"completed" : "N/A",
"request" : {
"type" : "DELETE",
```

```
"url" : "/scalemgmt/v2/smb/shares/share1/acl"
},
"result" : { },
"pids" : [ 15084 ]
} ],
"status" : {
"code" : 200,
"message" : "The request finished successfully"
}
}
```

Related reference

“mmsmb command” on page 709
Administers SMB shares, export ACLs, and global configuration.

SMB/shares/{shareName}/acl: GET

Gets information about access control lists (ACL) that are specific to an SMB share.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET smb/shares/shareName/acl` request gets information about the ACLs that are applicable to the specified SMB share. For more information about the fields in the data structures that are returned, see “mmsmb command” on page 709.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/smb/shares/shareName/acl
```

where

smb/shares
Specifies the SMB share as the resource. Required.

shareName/acl
Specifies the SMB share for which you need to get the ACL details. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 185. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
shareName	Name of the SMB share.	Required.

Request data

No request data.

Response data

The following list of attributes are available in the response data:

```
{
"status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   "paging":
    {
```

```
      "next":URL",
      "fields":"Fileds in request"
      "filter":"Filter in request"
      "baseUrl":"Base URL"
      "lastId":"ID"
       },
  "entries":
    {
        "shareName": "Share name"
        "name": "ACL name"
        "access": "Allowed | Denied"
        "permissions": "Access permissions",
        "type": "User | Group | System | UID",
    ),
```

The details of the parameters are given in the following list:

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
Paging is enabled when more than 1000 objects are returned by the query.

"next":*URL*"
The URL to retrieve the next page.

"fields":"*Fileds in the request*"
The fields used in the original request.

"filter":"*Filter in request*"
The filter used in the original request.

"baseUrl":"*Base URL*"
The URL of the request without any parameters.

"lastId":"*ID*"
The ID of the last element that can be used to retrieve the next elements.

"entries":
An array of information about an ACL that is applied to an SMB share.

"sharename": "*String*"
Name of the SMB share.

"name": "*ACL name*"
The name of the ACL.

"access": "*Allowed | Denied*"
Specifies whether the user is allowed or denied to access the resource.

"permissions": "*Access permissions*"
The permissions set for the ACL. Possible values are: FULL, CHANGE, READ or any combination of RWXDPO.

"type": "*User | Group | System | UID*"
Specifies the ACL type. The value selected is used only when creating an access control entry.

Examples

The following example gets information about the ACL of the SMB share `share1`.

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/smb/shares/share1/acl'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.


```
{
  "status": {
    "code": 200,
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/smb/shares/share1/acl?lastId=10001",
    "fields": "",
    "filter": "",
    "baseUrl": "smb/shares/share1/acl",
    "lastId": 10001
  },
  "entries": [
    {
      "shareName": "myshare",
      "name": "Domain1\\User3",
      "access": "ALLOWED",
      "permissions": "FULL",
      "type": "USER"
    }
  ]
}
```


Related reference

“mmsmb command” on page 709
Administers SMB shares, export ACLs, and global configuration.

SMB/shares/{shareName}/acl/{name}: DELETE

Deletes an entry from a specific access control list (ACL) that is applied to an SMB share.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `DELETE smb/shares/shareName/acl` command deletes the ACLs of the specified SMB share. For more information, see “mmsmb command” on page 709.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/smb/shares/shareName/acl/
name
```

where:

smb/shares
Specifies the SMB share as the resource. Required.

shareName/acl/name
Specifies the acl entry to be deleted. Required.

Request headers

```
Accept: application/json
```

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example deletes the ACL entry `Domain1\\User3`.

Request data:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https:///198.51.100.1:443/scalemgmt/v2/smb/shares/share1/acl/Domain1\\User3'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
"jobs" : [ {
"jobId" : 1000000000008,
"status" : "RUNNING",
"submitted" : "2017-08-24 04:45:34,034",
"completed" : "N/A",
"request" : {
"type" : "DELETE",
```

```
"url" : "/scalemgmt/v2/smb/shares/share1/acl/Domain1\\User3"
},
"result" : { },
"pids" : [ 15084 ]
} ],
"status" : {
"code" : 200,
"message" : "The request finished successfully"
}
}
```

Related reference

“mmsmb command” on page 709
Administers SMB shares, export ACLs, and global configuration.

SMB/shares/{shareName}/acl/{name}: GET

Gets information about a specific access control list (ACL) that is applicable to an SMB share.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET smb/shares/shareName/acl/name` request gets information about the ACLs that are applicable to the specified SMB share. For more information about the fields in the data structures that are returned, see “mmsmb command” on page 709.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/smb/shares/shareName/acl/
name
```

where

smb/shares
Specifies the SMB share as the resource. Required.

shareName/acl/name
Specifies the ACL about which you need the details. Required.

Request headers

```
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 186. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
shareName	Name of the SMB share.	Required.
name	Name of the ACL about which you need the details.	Required.

Request data

No request data.

Response data

The following list of attributes are available in the response data:

```
{
"status": {
      "code":ReturnCode",
```

```
      "message":"ReturnMessage"
   },
   "paging":
    {
      "next":URL",
      "fields":"Fileds in request"
      "filter":"Filter in request"
      "baseUrl":"Base URL"
      "lastId":"ID"
       },
  "entries":
    {
        "shareName": "Share name"
        "name": "ACL name"
        "access": "Allowed | Denied"
        "permissions": "Access permissions",
        "type": "User | Group | System | UID",
     ),
```

The details of the parameters are given in the following list:

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
Paging is enabled when more than 1000 objects are returned by the query.

"next":*URL*"
The URL to retrieve the next page.

"fields":"*Fileds in the request*"
The fields used in the original request.

"filter":"*Filter in request*"
The filter used in the original request.

"baseUrl":"*Base URL*"
The URL of the request without any parameters.

"lastId":"*ID*"
The ID of the last element that can be used to retrieve the next elements.

"entries":
An array of information about an ACL that is applied to an SMB share.

"sharename": "*String*"
Name of the SMB share.

"name": "*ACL name*"
The name of the ACL.

"access": "*Allowed | Denied*"
Specifies whether the user is allowed or denied to access the resource.

"permissions": "*Access permissions*"
The permissions set for the ACL. Possible values are: FULL, CHANGE, READ or any combination of RWXDPO.

"type": "*User | Group | System | UID*"
Specifies the ACL type. The value selected is used only when creating an access control entry.

Examples

The following example gets information about the ACL of the SMB share `share1`.

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/smb/shares/share1/acl/Domain1\\User3'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": 200,
    "message": "..."
  },
  "paging": {
    "next": "https://localhost:443/scalemgmt/v2/smb/shares/share1/acl?lastId=10001",
    "fields": "",
    "filter": "",
    "baseUrl": "smb/shares/share1/acl/Domain1\\User3",
    "lastId": 10001
  },
  "entries": [
    {
      "shareName": "share1",
      "name": "Domain1\\User3",
      "access": "ALLOWED",
      "permissions": "FULL",
      "type": "USER"
    }
  ]
}
```

Related reference

“mmsmb command” on page 709
Administers SMB shares, export ACLs, and global configuration.

SMB/shares/{shareName}/acl/{name}: PUT

Adds an entry to the ACL of a SMB share. If an entry for the specified name already exists, it gets replaced.

Note: The share name and the user name are part of both the URL and in the JSON payload. In JSON, both are optional, but if it is specified, they must match those in the URL.

Availability

Available on all IBM Spectrum Scale editions.

Description

The PUT `smb/shares/shareName/acl/name` request adds an entry to the ACL of a SMB share. For more information about the fields in the data structures that are returned, see “mmsmb command” on page 709.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/smb/shares/shareName/acl/
name
```

where

smb/shares
Specifies SMB share as the resource. Required.

shareName/acl/name
Specifies the ACL entry to be modified. Required.

Request headers

```
Accept: application/json
```

Request data

The following list of attributes are available in the response data:

```
 {
        "shareName": "Share name"
        "name": "ACL name"
        "access": "Allowed | Denied"
        "permissions": "Access permissions",
        "type": "User | Group | System | UID",
     )
```

The details of the parameters are given in the following list:

"sharename": "*String*"
Name of the SMB share.

"name": "*ACL name*"
The name of the ACL.

"access": "*Allowed | Denied*"
Specifies whether the user is allowed or denied to access the resource.

"permissions": "*Access permissions*"
The permissions set for the ACL. Possible values are: FULL, CHANGE, READ or any combination of RWXDPO.

"type": "*User | Group | System | UID*"
Specifies the ACL type. The value selected is used only when creating an access control entry.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
           {
             "type":"{GET | POST | PUT | DELETE}",
             "url":"URL",
             "data":"",
           }
         "jobId":"ID",
         "submitted":"Time",
         "completed":Time",
         "status":"Job status",
         }
       ],
   }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following example modifies the SMB share smbShare1.

Request data:

```
curl -k -u admin:admin001 -X PUT --header 'content-type:application/json' --header
'accept:application/json'
-d '{
   "shareName": "myshare",
   "name": "Domain1\\User3",
   "access": "ALLOWED",
   "permissions": "FULL",
   "type": "USER"   'https://198.51.100.1:443/scalemgmt/v2/smb/shares/smbShare1/acl/
Domain1%5C%5CUser3'
```

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {},
      "request": {
        "type": "PUT",
        "url": "/scalemgmt/v2/smb/shares/smbShare1/Domain1\\User3",
        "data": "{
           "shareName": "myshare",
           "name": "Domain1\\User3",
           "access": "ALLOWED",
           "permissions": "FULL",
           "type": "USER"
      }",
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

Related reference

“mmsmb command” on page 709
Administers SMB shares, export ACLs, and global configuration.

Thresholds: GET

Gets a list of all threshold rules that are configured in the IBM Spectrum Scale cluster.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET thresholds` request gets a list of all threshold rules that are configured in the system. For more information about the fields in the data structures that are returned, see “mmhealth command” on page 441.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/thresholds
```

where

thresholds
: Specifies the threshold rules that are configured in the cluster as the resource of this GET call. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request data

No request data.

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 187. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.

Response data

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "thresholds": [
    {
```

```
      "ruleName": "Name",
      "frequency": "Frequency",
      "tags": "Tags",
      "userActionWarn": "Warning message",
      "userActionError": "Error message",
      "type": "metric",
      "metric": "Metric name",
      "metricOp": "Metric operation",
      "sensitivity": "Sensitivity",
      "computation": "value",
      "duration": "Duration",
      "filterBy": "Filter by",
      "groupBy": "Group by",
      "errorLevel": "Error level",
      "warnLevel": "Warning level",
      "direction": "Direction",
      "hysteresis": "Hysterisis"
    }
  ]
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"thresholds":

"ruleName":"*Name*"
The name of the threshold rule.

"frequency":"*Time*"
The frequency in which the threshold values are evaluated.

"tags": "*tags*"
Maps events to components such as pool_data, pool_meta-data, fset_inode, and thresholds.

"userActionWarn": "*Warning message*"
A user-defined message that is included in the warning message.

"userActionError": "*Error message*"
A user-defined message that is included in the error message.

"type": "*metric | measurement*"
The type of the threshold, either metric or measurement.

"metric": "*Metric name*"
The metric for which the threshold rule is defined.

"metricOp": "*sum | avg | min | max | rate*"
The metric operation or the aggregator that is used for the threshold rule.

"sensitivity": "*Sensitivity*"
The sample interval value in seconds.

"computation": "*Computation criteria*"
A rule consists of a computation element performs a computation on the collected data. There are four computation criteria defined: value, stats (max, min, median, and percentile), zimonstatus, and gpfsCapacityUtil. Currently, only 'value' is supported.

"duration": "*Collection duration*"
Duration of collection time (in seconds).

"filterBy": "*Filter by*"
Filters the result based on the filter key.

"groupBy": "*Group by*"
Groups the result based on the group key.

"errorLevel": "*Error level*"
The threshold error limit that is defined for the metric. The value can be a percentage or an integer, depending on the metric on which the threshold value is being set.

"warnLevel": "*Warning level*"
The threshold warning limit that is defined for the metric. The value can be a percentage or an integer, depending on the metric on which the threshold value is being set.

"direction": "*High | Low*"
The direction for the threshold limit.

"hysteresis": "*Hysterisis*"
The percentage that the observed value must be above or below the current threshold level to switch back to the previous state.

Examples

The following example gets the list of threshold rules that are defined in the system.

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/thresholds?fields=:all:'
```

Note: The field ':all:' selects all available fields.

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "thresholds": [
    {
      "ruleName": "string",
      "frequency": "300",
      "tags": "300",
      "userActionWarn": "The cpu usage has exceeded the warning level. ",
      "userActionError": "The cpu usage has exceeded the threshold level. ",
      "type": "metric",
      "metric": "cpu_user",
      "metricOp": "avg",
      "sensitivity": "300",
      "computation": "value",
      "duration": "300",
      "filterBy": "gpfs_fs_name=gpfs0",
      "groupBy": "gpfs_fs_name=gpfs0",
      "errorLevel": "90.0",
      "warnLevel": "80.0",
      "direction": "high",
      "hysteresis": "5.0"
    }
  ]
}
```

Related reference

“mmhealth command” on page 441
Monitors health status of nodes.

Thresholds: POST

Create a threshold rule.

Availability

Available on all IBM Spectrum Scale editions.

Description

The POST `thresholds` request creates a threshold rule that defines threshold levels for data that is collected through performance monitoring sensors. For more information about the fields in the data structures that are returned, see "mmhealth command" on page 441.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/thresholds
```

where

thresholds
Specifies thresholds as the target of the operation. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 188. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
body	Body of the request that contains the required parameters to be passed on to the IBM Spectrum Scale system to perform the requested operation.	Required.

Request data

```
{
    "ruleName": "Name",
    "frequency": "Frequency",
    "tags": "Tags",
    "userActionWarn": "Warning message",
    "userActionError": "Error message",
    "type": "metric",
    "metric": "Metric name",
    "metricOp": "Metric operation",
    "sensitivity": "Sensitivity",
    "computation": "value",
    "duration": "Duration",
    "filterBy": "Filter by",
    "groupBy": "Group by",
    "errorLevel": "Error level",
    "warnLevel": "Warning level",
    "direction": "Direction",
```

```
      "hysteresis": "Hysterisis"
    }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"ruleName":"*Name*"
The name of the threshold rule.

"frequency":"*Time*"
The frequency in which the threshold values are evaluated.

"tags": "*tags*"
Maps events to components such as pool_data, pool_meta-data, fset_inode, and thresholds.

"userActionWarn": "*Warning message*"
A user-defined message that is included in the warning message.

"userActionError": "*Error message*"
A user-defined message that is included in the error message.

"type": "*metric | measurement*"
The type of the threshold, either metric or measurement.

"metric": "*Metric name*"
The metric for which the threshold rule is defined.

"metricOp": "*sum | avg | min | max | rate*"
The metric operation or the aggregator that is used for the threshold rule.

"sensitivity": "*Sensitivity*"
The sample interval value in seconds.

"computation": "*Computation criteria*"
A rule consists of a computation element performs a computation on the collected data. There are four computation criteria defined: value, stats (max, min, median, and percentile), zimonstatus, and gpfsCapacityUtil. Currently, only 'value' is supported.

"duration": "*Collection duration*"
Duration of collection time (in seconds).

"filterBy": "*Filter by*"
Filters the result based on the filter key.

"groupBy": "*Group by*"
Groups the result based on the group key.

"errorLevel": "*Error level*"
The threshold error limit that is defined for the metric. The value can be a percentage or an integer, depending on the metric on which the threshold value is being set.

"warnLevel": "*Warning level*"
The threshold warning limit that is defined for the metric. The value can be a percentage or an integer, depending on the metric on which the threshold value is being set.

"direction": "*High | Low*"
The direction for the threshold limit.

"hysteresis": "*Hysterisis*"
The percentage that the observed value must be above or below the current threshold level to switch back to the previous state.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
```

```
        "result":"",
          {
            "commands":"String",
            "progress":"String,
            "exitCode":"Exit code",
            "stderr":"Error",
            "stdout":"String",
          },
        "request":" ",
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted":"Time",
        "completed":Time",
        "status":"Job status",
        }
       ],
  }
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following API command creates a threshold rule *rule1*.

Request URL:

```
curl -k -u admin:admin001 -X POST --header 'content-type:application/json' --header
'accept:application/json'
-d "{
  "ruleName": "rule1",
  "userActionWarn": "The cpu usage has exceeded the warning level. ",
  "userActionError": "The cpu usage has exceeded the threshold level. ",
  "metric": "cpu_user",
  "sensitivity": "300",
  "filterBy": "gpfs_fs_name=gpfs0",
  "groupBy": "gpfs_fs_name=gpfs0",
  "errorLevel": "90.0",
  "warnLevel": "80.0",
  "direction": "high",
  "hysteresis": "5.0"
}
'https://198.51.100.1:443/scalemgmt/v2/thresholds'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": ["'mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": ["'(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": ["'EFSSG0740C There are not enough resources available to create
                  a new independent file set.'',  ...]",
        "stdout": ["'EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "POST",
        "url": "/scalemgmt/v2/thresholds",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

Related reference

"mmhealth command" on page 441
Monitors health status of nodes.

Thresholds/{name}: DELETE

Deletes a specific threshold rule.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `DELETE thresholds` request deletes a specific threshold rule. A threshold rule defines threshold levels for data that is collected through performance monitoring sensors. For more information about the fields in the data structures that are returned, see “mmhealth command” on page 441.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/thresholds/name
```

where

thresholds/name
Specifies thresholds as the target of the operation. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 189. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
name	The name of the threshold rule to be deleted.	Required.

Request data

No request data.

Response data

```
{
    "status": {
      "code":ReturnCode",
      "message":"ReturnMessage"
   },
   jobs: [
      {
         "result":"",
           {
             "commands":"String",
             "progress":"String,
             "exitCode":"Exit code",
             "stderr":"Error",
             "stdout":"String",
           },
         "request":" ",
```

```
          {
            "type":"{GET | POST | PUT | DELETE}",
            "url":"URL",
            "data":"",
          }
        "jobId":"ID",
        "submitted":"Time",
        "completed":Time",
        "status":"Job status",
        }
      ],
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"paging"
The URL to retrieve the next page. Paging is enabled when more than 1000 objects are returned by the query.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"jobs":
An array of elements that describe jobs. Each element describes one job.

"result"

"commands":"*String*"
Array of commands that are run in this job.

"progress":"*String*"
Progress information for the request.

"exitCode":"*Exit code*"
Exit code of command. Zero is success, nonzero denotes failure.

"stderr":"*Error*"
CLI messages from *stderr.*

"stdout":"*String*"
CLI messages from stdout.

"request"

"type":"{GET | POST | PUT | DELETE}"
HTTP request type.

"url":"*URL*"
The URL through which the job is submitted.

"data":" "
Optional.

"jobId":"*ID*",
The unique ID of the job.

"submitted":"*Time*"
The time at which the job was submitted.

"completed":*Time*"
The time at which the job was completed.

"status":"RUNNING | COMPLETED | FAILED"
Status of the job.

Examples

The following API command deletes the threshold rule *rule1*.

Request URL:

```
curl -k -u admin:admin001 -X DELETE --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/thresholds/rule1'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "job": [
    {
      "result": {
        "commands": "[''mmcrfileset gpfs0 restfs1001'', ...]",
        "progress": "[''(2/3) Linking fileset'']",
        "exitCode": "0",
        "stderr": "[''EFSSG0740C There are not enough resources available to create
                  a new independent file set.'',  ...]",
        "stdout": "[''EFSSG4172I The file set {0} must be independent.'', ...]"
      },
      "request": {
        "type": "DELETE",
        "url": "/scalemgmt/v2/thresholds/rule1",
        "data":  "nodesDesc": "[ 'mari-16:manager-quorum', 'mari-17::mari-17_admin' ]"
      },
      "jobId": "12345",
      "submitted": "2016-11-14 10.35.56",
      "completed": "2016-11-14 10.35.56",
      "status": "COMPLETED"
    }
  ]
}
```

Related reference

“mmhealth command” on page 441
Monitors health status of nodes.

Thresholds/{name}: GET

Gets details of a specific threshold rule that is configured in the IBM Spectrum Scale cluster.

Availability

Available on all IBM Spectrum Scale editions.

Description

The `GET thresholds/{name}` request gets details of the specified threshold rule. For more information about the fields in the data structures that are returned, see “mmhealth command” on page 441.

Request URL

```
https://<IP address or host name of API server>:<port>/scalemgmt/v2/thresholds/name
```

where

thresholds/name
Specifies the threshold rule about which you need the details. Required.

Request headers

```
Content-Type: application/json
Accept: application/json
```

Request data

No request data.

Request parameters

The following parameters can be used in the request URL to customize the request:

Table 190. List of request parameters

Parameter name	Description and applicable keywords	Required/optional
fields	Comma separated list of fields to be included in response. ':all:' selects all available fields.	Optional.
filter	Filter objects by expression. For example, 'status=HEALTHY,entityType=FILESET'	Optional.
name	The name of the threshold rule.	Required.

Response data

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "thresholds": [
    {
      "ruleName": "Name",
```

```
      "frequency": "Frequency",
      "tags": "Tags",
      "userActionWarn": "Warning message",
      "userActionError": "Error message",
      "type": "metric",
      "metric": "Metric name",
      "metricOp": "Metric operation",
      "sensitivity": "Sensitivity",
      "computation": "value",
      "duration": "Duration",
      "filterBy": "Filter by",
      "groupBy": "Group by",
      "errorLevel": "Error level",
      "warnLevel": "Warning level",
      "direction": "Direction",
      "hysteresis": "Hysterisis"
    }
  ]
}
```

For more information about the fields in the following data structures, see the links at the end of this topic.

"status":
Return status.

"message": "*ReturnMessage*",
The return message.

"code": *ReturnCode*
The return code.

"thresholds":

"ruleName":"*Name*"
The name of the threshold rule.

"frequency":"*Time*"
The frequency in which the threshold values are evaluated.

"tags": "*tags*"
Maps events to components such as pool_data, pool_meta-data, fset_inode, and thresholds.

"userActionWarn": "*Warning message*"
A user-defined message that is included in the warning message.

"userActionError": "*Error message*"
A user-defined message that is included in the error message.

"type": "*metric | measurement*"
The type of the threshold, either metric or measurement.

"metric": "*Metric name*"
The metric for which the threshold rule is defined.

"metricOp": "*sum | avg | min | max | rate*"
The metric operation or the aggregator that is used for the threshold rule.

"sensitivity": "*Sensitivity*"
The sample interval value in seconds.

"computation": "*Computation criteria*"
A rule consists of a computation element performs a computation on the collected data. There are four computation criteria defined: value, stats (max, min, median, and percentile), zimonstatus, and gpfsCapacityUtil. Currently, only 'value' is supported.

"duration": "*Collection duration*"
Duration of collection time (in seconds).

"filterBy": "*Filter by*"
Filters the result based on the filter key.

"groupBy": "*Group by*"
Groups the result based on the group key.

"errorLevel": "*Error level*"
The threshold error limit that is defined for the metric. The value can be a percentage or an integer, depending on the metric on which the threshold value is being set.

"warnLevel": "*Warning level*"
The threshold warning limit that is defined for the metric. The value can be a percentage or an integer, depending on the metric on which the threshold value is being set.

"direction": "*High | Low*"
The direction for the threshold limit.

"hysteresis": "*Hysterisis*"
The percentage that the observed value must be above or below the current threshold level to switch back to the previous state.

Examples

The following example gets details of the threshold rule *rule1*

Request URL:

```
curl -k -u admin:admin001 -X GET --header 'accept:application/json'
'https://198.51.100.1:443/scalemgmt/v2/thresholds/rule1'
```

The request URL with no field or filter parameter returns only the details that uniquely identify the object.

Response data:

Note: In the JSON data that is returned, the return code indicates whether the command is successful. The response code 200 indicates that the command successfully retrieved the information. Error code 400 represents an invalid request and 500 represents internal server error.

```
{
  "status": {
    "code": "200",
    "message": "..."
  },
  "thresholds": [
    {
      "ruleName": "rule1",
      "frequency": "300",
      "tags": "300",
      "userActionWarn": "The cpu usage has exceeded the warning level. ",
      "userActionError": The cpu usage has exceeded the threshold level.",
      "type": "metric",
      "metric": "cpu_user",
      "metricOp": "avg",
      "sensitivity": "300",
      "computation": "value",
      "duration": "300",
      "filterBy": "gpfs_fs_name=gpfs0",
      "groupBy": "gpfs_fs_name=gpfs0",
      "errorLevel": "90.0",
      "warnLevel": "80.0",
      "direction": "high",
      "hysteresis": "5.0"
    }
  ]
}
```

Related reference

“mmhealth command” on page 441
Monitors health status of nodes.

Chapter 6. Considerations for GPFS applications

Application design must consider the exceptions to the Open Group technical standards for the `stat()` system call. Also, a technique to check if a file system is controlled by GPFS has been provided.

Exceptions to Open Group technical standards

GPFS is designed in a way that most applications written to the Open Group technical standard for file system calls can access the GPFS data without any modification. However, there are some exceptions.

Applications that depend on exact reporting of changes to the following fields returned by the `stat()` call may not work as expected:

1. `exact mtime`
2. `mtime`
3. `ctime`
4. `atime`

Providing exact support for these fields would require significant performance degradation to all applications executing on the system. These fields are guaranteed accurate when the file is closed.

These values will be accurate on a node right after it accesses or modifies a file, but may not be accurate for a short while when a file is accessed or modified on some other node.

If 'exact mtime' is specified for a file system (using the `mmcrfs` or `mmchfs` commands with the `-E yes` flag), the `mtime` and `ctime` values are always correct by the time the `stat()` call gives its answer.
If 'exact mtime' is not specified, these values will be accurate after a couple of minutes, to allow the synchronization daemons to propagate the values to all nodes. Regardless of whether 'exact mtime' is specified, the `atime` value will be accurate after a couple of minutes, to allow for all the synchronization daemons to propagate changes.

Alternatively, you may use the GPFS calls, `gpfs_stat()` and `gpfs_fstat()` to return exact `mtime` and `atime` values.

The delayed update of the information returned by the `stat()` call also impacts system commands which display disk usage, such as `du` or `df`. The data reported by such commands may not reflect changes that have occurred since the last sync of the file system. For a parallel file system, a sync does not occur until all nodes have individually synchronized their data. On a system with no activity, the correct values will be displayed after the sync daemon has run on all nodes.

Determining if a file system is controlled by GPFS

A file system is controlled by GPFS if the `f_type` field in the `statfs` structure returned from a `statfs()` or `fstatfs()` call has the value 0x47504653, which is GPFS in ASCII.

About this task

This constant is in the `gpfs.h` file, with the name `GPFS_SUPER_MAGIC`. If an application includes `gpfs.h`, it can compare `f_type` to `GPFS_SUPER_MAGIC` to determine if the file system is controlled by GPFS.

Considerations for the use of direct I/O (O_DIRECT)

When a file is opened in the **`O_DIRECT`** mode (direct I/O mode), GPFS transfers data directly between the user buffer and the file on the disk.

Using direct I/O may provide some performance benefits in the following cases:

- The file is accessed at random locations.
- There is no access locality.

Direct transfer between the user buffer and the disk can only happen if all of the following conditions are true:

- The number of bytes transferred is a multiple of 512 bytes.
- The file offset is a multiple of 512 bytes.
- The user memory buffer address is aligned on a 512-byte boundary.

When any of these conditions is false, the operation proceeds but is treated more like other normal file I/O, with the **O_SYNC** flag that flushes the dirty buffer to disk.

When these conditions are all true, the GPFS page pool is not used because the data is transferred directly. Therefore, an environment in which most of the I/O volume is due to direct I/O (such as in databases) does not benefit from a large page pool. However, that the page pool still needs to be configured to an adequate size or left at its default value. The reason is that the page pool is also used to store file metadata, especially for the indirect blocks required for large files.

In IBM Spectrum Scale 5.0.5 and later, the configuration parameter **minIndBlkDescs** can be used to set the cache size for indirect blocks. The applications that perform direct I/O on very large files might benefit from this parameter. By specifying a large value for **minIndBlkDescs**, you can ensure that more indirect block descriptors are cached in memory. For more information, see “mmchconfig command” on page 170.

When direct I/O is done on 4K sector disks, the alignment requirement for the number of bytes, the file offset, and the user memory buffer is 4K bytes instead of 512 bytes. If such an alignment is not in place during an individual direct I/O operation, the request is then treated like normal I/O, as explained earlier.

With direct I/O, the application is responsible for coordinating access to the file, and neither the overhead nor the protection provided by GPFS locking mechanisms plays a role. In particular, if two threads or nodes perform direct I/O concurrently on overlapping portions of the file, the outcome is undefined. For example, when multiple writes are made to the same file offsets, it is undetermined which of the writes will be on the file when all I/O is completed. In addition, if the file has data replication, it is not guaranteed that all the data replicas will contain the data from the same writer. That is, the contents of each of the replicas could diverge.

Even when the I/O requests are aligned as previously listed, in the following cases GPFS will not transfer the data directly and will revert to the slower buffered behavior:

- The write causes the file to increase in size.
- The write is in a region of the file that has been per-allocated (via **gpfs_prealloc()**) but has not yet been written.
- The write is in a region of the file where a "hole" is present; that is, the file is sparse and has some unallocated regions.

When direct I/O requests are aligned but none of the previously listed conditions (that would cause the buffered I/O path to be taken) are present, handling is optimized this way: the request is completely handled in kernel mode by the GPFS kernel module, without the GPFS daemon getting involved. Any of the following conditions, however, will still result in the request going through the daemon:

- The I/O operation needs to be served by an NSD server.
- The file system has data replication. In the case of a write operation, the GPFS daemon is involved to produce the log records that ensure that the replica contents are identical (in case of a failure while writing the replicas to disk).
- The operation is performed on the Windows operating system.

Note that setting the **O_DIRECT** flag on an open file with **fcntl (fd, F_SETFL,[..])**, which may be allowed on Linux, is ignored in a GPFS file system.

Because of a limitation in Linux, I/O operations with **O_DIRECT** should not be issued concurrently with a **fork(2)** system call that is invoked by the same process. Any calls to **fork()** in the program should be

issued only after **O_DIRECT** I/O operations are completed. That is, **fork()** should not be invoked while **O_DIRECT** I/O operations are still pending completion. For more information, see the **open(2)** system call in the Linux documentation.

Accessibility features for IBM Spectrum Scale

Accessibility features help users who have a disability, such as restricted mobility or limited vision, to use information technology products successfully.

Accessibility features

The following list includes the major accessibility features in IBM Spectrum Scale:

- Keyboard-only operation
- Interfaces that are commonly used by screen readers
- Keys that are discernible by touch but do not activate just by touching them
- Industry-standard devices for ports and connectors
- The attachment of alternative input and output devices

IBM Documentation, and its related publications, are accessibility-enabled.

Keyboard navigation

This product uses standard Microsoft Windows navigation keys.

IBM and accessibility

See the IBM Human Ability and Accessibility Center (www.ibm.com/able) for more information about the commitment that IBM has to accessibility.

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing IBM Corporation North Castle Drive, MD-NC119 Armonk, NY 10504-1785 US

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing Legal and Intellectual Property Law IBM Japan Ltd. 19-21, Nihonbashi-Hakozakicho, Chuo-ku Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing IBM Corporation North Castle Drive, MD-NC119 Armonk, NY 10504-1785 US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and

cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (your company name) (year).
Portions of this code are derived from IBM Corp.
Sample Programs. © Copyright IBM Corp. _enter the year or years_.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

Intel is a trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Red Hat®, OpenShift®, and Ansible are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of the Open Group in the United States and other countries.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, See IBM’s Privacy Policy at http://www.ibm.com/privacy and IBM’s Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled “Cookies, Web Beacons and Other Technologies” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at http://www.ibm.com/software/info/product-privacy.

Glossary

This glossary provides terms and definitions for IBM Spectrum Scale.

The following cross-references are used in this glossary:

- *See* refers you from a nonpreferred term to the preferred term or from an abbreviation to the spelled-out form.
- *See also* refers you to a related or contrasting term.

For other terms and definitions, see the IBM Terminology website (www.ibm.com/software/globalization/terminology) (opens in new window).

B

block utilization
The measurement of the percentage of used subblocks per allocated blocks.

C

cluster
A loosely coupled collection of independent systems (nodes) organized into a network for the purpose of sharing resources and communicating with each other. See also *GPFS cluster*.

cluster configuration data
The configuration data that is stored on the cluster configuration servers.

Cluster Export Services (CES) nodes
A subset of nodes configured within a cluster to provide a solution for exporting GPFS file systems by using the Network File System (NFS), Server Message Block (SMB), and Object protocols.

cluster manager
The node that monitors node status using disk leases, detects failures, drives recovery, and selects file system managers. The cluster manager must be a quorum node. The selection of the cluster manager node favors the quorum-manager node with the lowest node number among the nodes that are operating at that particular time.

Note: The cluster manager role is not moved to another node when a node with a lower node number becomes active.

clustered watch folder
Provides a scalable and fault-tolerant method for file system activity within an IBM Spectrum Scale file system. A clustered watch folder can watch file system activity on a fileset, inode space, or an entire file system. Events are streamed to an external Kafka sink cluster in an easy-to-parse JSON format. For more information, see the *mmwatch command* in the *IBM Spectrum Scale: Command and Programming Reference*.

control data structures
Data structures needed to manage file data and metadata cached in memory. Control data structures include hash tables and link pointers for finding cached data; lock states and tokens to implement distributed locking; and various flags and sequence numbers to keep track of updates to the cached data.

D

Data Management Application Program Interface (DMAPI)
The interface defined by the Open Group's XDSM standard as described in the publication *System Management: Data Storage Management (XDSM) API Common Application Environment (CAE) Specification C429*, The Open Group ISBN 1-85912-190-X.

deadman switch timer
A kernel timer that works on a node that has lost its disk lease and has outstanding I/O requests. This timer ensures that the node cannot complete the outstanding I/O requests (which would risk causing file system corruption), by causing a panic in the kernel.

dependent fileset
A fileset that shares the inode space of an existing independent fileset.

disk descriptor
A definition of the type of data that the disk contains and the failure group to which this disk belongs. See also *failure group*.

disk leasing
A method for controlling access to storage devices from multiple host systems. Any host that wants to access a storage device configured to use disk leasing registers for a lease; in the event of a perceived failure, a host system can deny access, preventing I/O operations with the storage device until the preempted system has reregistered.

disposition
The session to which a data management event is delivered. An individual disposition is set for each type of event from each file system.

domain
A logical grouping of resources in a network for the purpose of common management and administration.

E

ECKD
See *extended count key data (ECKD)*.

ECKD device
See *extended count key data device (ECKD device)*.

encryption key
A mathematical value that allows components to verify that they are in communication with the expected server. Encryption keys are based on a public or private key pair that is created during the installation process. See also *file encryption key*, *master encryption key*.

extended count key data (ECKD)
An extension of the count-key-data (CKD) architecture. It includes additional commands that can be used to improve performance.

extended count key data device (ECKD device)
A disk storage device that has a data transfer rate faster than some processors can utilize and that is connected to the processor through use of a speed matching buffer. A specialized channel program is needed to communicate with such a device. See also *fixed-block architecture disk device*.

F

failback
Cluster recovery from failover following repair. See also *failover*.

failover
(1) The assumption of file system duties by another node when a node fails. (2) The process of transferring all control of the ESS to a single cluster in the ESS when the other clusters in the ESS fails. See also *cluster*. (3) The routing of all transactions to a second controller when the first controller fails. See also *cluster*.

failure group
A collection of disks that share common access paths or adapter connections, and could all become unavailable through a single hardware failure.

FEK
See *file encryption key*.

fileset
A hierarchical grouping of files managed as a unit for balancing workload across a cluster. See also *dependent fileset*, *independent fileset*.

fileset snapshot
A snapshot of an independent fileset plus all dependent filesets.

file audit logging
Provides the ability to monitor user activity of IBM Spectrum Scale file systems and store events related to the user activity in a security-enhanced fileset. Events are stored in an easy-to-parse JSON format. For more information, see the *mmaudit command* in the *IBM Spectrum Scale: Command and Programming Reference*.

file clone
A writable snapshot of an individual file.

file encryption key (FEK)
A key used to encrypt sectors of an individual file. See also *encryption key*.

file-management policy
A set of rules defined in a policy file that GPFS uses to manage file migration and file deletion. See also *policy*.

file-placement policy
A set of rules defined in a policy file that GPFS uses to manage the initial placement of a newly created file. See also *policy*.

file system descriptor
A data structure containing key information about a file system. This information includes the disks assigned to the file system (*stripe group*), the current state of the file system, and pointers to key files such as quota files and log files.

file system descriptor quorum
The number of disks needed in order to write the file system descriptor correctly.

file system manager
The provider of services for all the nodes using a single file system. A file system manager processes changes to the state or description of the file system, controls the regions of disks that are allocated to each node, and controls token management and quota management.

fixed-block architecture disk device (FBA disk device)
A disk device that stores data in blocks of fixed size. These blocks are addressed by block number relative to the beginning of the file. See also *extended count key data device*.

fragment
The space allocated for an amount of data too small to require a full block. A fragment consists of one or more subblocks.

G

global snapshot
A snapshot of an entire GPFS file system.

GPFS cluster
A cluster of nodes defined as being available for use by GPFS file systems.

GPFS portability layer
The interface module that each installation must build for its specific hardware platform and Linux distribution.

GPFS recovery log
A file that contains a record of metadata activity and exists for each node of a cluster. In the event of a node failure, the recovery log for the failed node is replayed, restoring the file system to a consistent state and allowing other nodes to continue working.

I

ill-placed file
A file assigned to one storage pool but having some or all of its data in a different storage pool.

ill-replicated file
A file with contents that are not correctly replicated according to the desired setting for that file. This situation occurs in the interval between a change in the file's replication settings or suspending one of its disks, and the restripe of the file.

independent fileset
A fileset that has its own inode space.

indirect block
A block containing pointers to other blocks.

inode
The internal structure that describes the individual files in the file system. There is one inode for each file.

inode space
A collection of inode number ranges reserved for an independent fileset, which enables more efficient per-fileset functions.

ISKLM
IBM Security Key Lifecycle Manager. For GPFS encryption, the ISKLM is used as an RKM server to store MEKs.

J

journaled file system (JFS)
A technology designed for high-throughput server environments, which are important for running intranet and other high-performance e-business file servers.

junction
A special directory entry that connects a name in a directory of one fileset to the root directory of another fileset.

K

kernel
The part of an operating system that contains programs for such tasks as input/output, management and control of hardware, and the scheduling of user tasks.

M

master encryption key (MEK)
A key used to encrypt other keys. See also *encryption key*.

MEK
See *master encryption key*.

metadata
Data structures that contain information that is needed to access file data. Metadata includes inodes, indirect blocks, and directories. Metadata is not accessible to user applications.

metanode
The one node per open file that is responsible for maintaining file metadata integrity. In most cases, the node that has had the file open for the longest period of continuous time is the metanode.

mirroring
The process of writing the same data to multiple disks at the same time. The mirroring of data protects it against data loss within the database or within the recovery log.

Microsoft Management Console (MMC)

A Windows tool that can be used to do basic configuration tasks on an SMB server. These tasks include administrative tasks such as listing or closing the connected users and open files, and creating and manipulating SMB shares.

multi-tailed
A disk connected to multiple nodes.

N

namespace
Space reserved by a file system to contain the names of its objects.

Network File System (NFS)
A protocol, developed by Sun Microsystems, Incorporated, that allows any host in a network to gain access to another host or netgroup and their file directories.

Network Shared Disk (NSD)
A component for cluster-wide disk naming and access.

NSD volume ID
A unique 16-digit hex number that is used to identify and access all NSDs.

node
An individual operating-system image within a cluster. Depending on the way in which the computer system is partitioned, it may contain one or more nodes.

node descriptor
A definition that indicates how GPFS uses a node. Possible functions include: manager node, client node, quorum node, and nonquorum node.

node number
A number that is generated and maintained by GPFS as the cluster is created, and as nodes are added to or deleted from the cluster.

node quorum
The minimum number of nodes that must be running in order for the daemon to start.

node quorum with tiebreaker disks
A form of quorum that allows GPFS to run with as little as one quorum node available, as long as there is access to a majority of the quorum disks.

non-quorum node
A node in a cluster that is not counted for the purposes of quorum determination.

Non-Volatile Memory Express (NVMe)
An interface specification that allows host software to communicate with non-volatile memory storage media.

P

policy
A list of file-placement, service-class, and encryption rules that define characteristics and placement of files. Several policies can be defined within the configuration, but only one policy set is active at one time.

policy rule
A programming statement within a policy that defines a specific action to be performed.

pool
A group of resources with similar characteristics and attributes.

portability
The ability of a programming language to compile successfully on different operating systems without requiring changes to the source code.

primary GPFS cluster configuration server
In a GPFS cluster, the node chosen to maintain the GPFS cluster configuration data.

private IP address
An IP address used to communicate on a private network.

public IP address
An IP address used to communicate on a public network.

Q

quorum node
A node in the cluster that is counted to determine whether a quorum exists.

quota
The amount of disk space and number of inodes assigned as upper limits for a specified user, group of users, or fileset.

quota management
The allocation of disk blocks to the other nodes writing to the file system, and comparison of the allocated space to quota limits at regular intervals.

R

Redundant Array of Independent Disks (RAID)
A collection of two or more disk physical drives that present to the host an image of one or more logical disk drives. In the event of a single physical device failure, the data can be read or regenerated from the other disk drives in the array due to data redundancy.

recovery
The process of restoring access to file system data when a failure has occurred. Recovery can involve reconstructing data or providing alternative routing through a different server.

remote key management server (RKM server)
A server that is used to store master encryption keys.

replication
The process of maintaining a defined set of data in more than one location. Replication consists of copying designated changes for one location (a source) to another (a target) and synchronizing the data in both locations.

RKM server
See *remote key management server*.

rule
A list of conditions and actions that are triggered when certain conditions are met. Conditions include attributes about an object (file name, type or extension, dates, owner, and groups), the requesting client, and the container name associated with the object.

S

SAN-attached
Disks that are physically attached to all nodes in the cluster using Serial Storage Architecture (SSA) connections or using Fibre Channel switches.

Scale Out Backup and Restore (SOBAR)
A specialized mechanism for data protection against disaster only for GPFS file systems that are managed by IBM Spectrum Protect Hierarchical Storage Management (HSM).

secondary GPFS cluster configuration server
In a GPFS cluster, the node chosen to maintain the GPFS cluster configuration data in the event that the primary GPFS cluster configuration server fails or becomes unavailable.

Secure Hash Algorithm digest (SHA digest)
A character string used to identify a GPFS security key.

session failure
The loss of all resources of a data management session due to the failure of the daemon on the session node.

session node
The node on which a data management session was created.

Small Computer System Interface (SCSI)
An ANSI-standard electronic interface that allows personal computers to communicate with peripheral hardware, such as disk drives, tape drives, CD-ROM drives, printers, and scanners faster and more flexibly than previous interfaces.

snapshot
An exact copy of changed data in the active files and directories of a file system or fileset at a single point in time. See also *fileset snapshot*, *global snapshot*.

source node
The node on which a data management event is generated.

stand-alone client
The node in a one-node cluster.

storage area network (SAN)
A dedicated storage network tailored to a specific environment, combining servers, storage products, networking products, software, and services.

storage pool
A grouping of storage space consisting of volumes, logical unit numbers (LUNs), or addresses that share a common set of administrative characteristics.

stripe group
The set of disks comprising the storage assigned to a file system.

striping
A storage process in which information is split into blocks (a fixed amount of data) and the blocks are written to (or read from) a series of disks in parallel.

subblock
The smallest unit of data accessible in an I/O operation, equal to one thirty-second of a data block.

system storage pool
A storage pool containing file system control structures, reserved files, directories, symbolic links, special devices, as well as the metadata associated with regular files, including indirect blocks and extended attributes. The `system storage pool` can also contain user data.

T

token management
A system for controlling file access in which each application performing a read or write operation is granted some form of access to a specific block of file data. Token management provides data consistency and controls conflicts. Token management has two components: the token management server, and the token management function.

token management function
A component of token management that requests tokens from the token management server. The token management function is located on each cluster node.

token management server
A component of token management that controls tokens relating to the operation of the file system. The token management server is located at the file system manager node.

transparent cloud tiering (TCT)
A separately installable add-on feature of IBM Spectrum Scale that provides a native cloud storage tier. It allows data center administrators to free up on-premise storage capacity, by moving out cooler data to the cloud storage, thereby reducing capital and operational expenditures.

twin-tailed
A disk connected to two nodes.

U

user storage pool
A storage pool containing the blocks of data that make up user files.

V

VFS
See *virtual file system*.

virtual file system (VFS)
A remote file system that has been mounted so that it is accessible to the local user.

virtual node (vnode)
The structure that contains information about a file system object in a virtual file system (VFS).

W

watch folder API
Provides a programming interface where a custom C program can be written that incorporates the ability to monitor inode spaces, filesets, or directories for specific user activity-related events within IBM Spectrum Scale file systems. For more information, a sample program is provided in the following directory on IBM Spectrum Scale nodes: `/usr/lpp/mmfs/samples/util` called tswf that can be modified according to the user's needs.

Index

Special Characters

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Product Number: 5641-DM1
5641-DM3
5641-DM5
5641-DA1
5641-DA3
5641-DA5
5737-F34
5737-I39
5765-DME
5765-DAE

GC28-3197-02